A Personalization Framework to Improve Quality of Experience for DVD-like Functions in P2P VoD Applications

Tianyin Xu*‡, Baoliu Ye*, Qinhui Wang*, Wenzhong Li*, Sanglu Lu*, Xiaoming Fu‡

*State Key Lab. for Novel Software and Technology, Nanjing University, China

‡Institute of Computer Science, University of Goettingen, Germany

Abstract

The requirement for supporting DVD-like functions raises new challenges to the design of P2P VoD systems. The uncertainty of frequent user DVD-like interactivity makes it difficult to ensure user perceived Quality of Experience (QoE) for real-time streaming services over distributed self-organized P2P overlay networks. Most existing solutions are based on the unreasonable assumption that all the users in P2P VoD systems have the same preference. Few attention has been paid to personalization, which accommodates the differences between users. In this paper, we present a video model which characterizes the personalization information for users' contents and preferences. Based on this model, we develop APEX, a practical personalization framework for P2P VoD applications. APEX makes the personalization practical by using a hybrid architecture which leverages the offline pattern mining on the server side and online collaborative filtering on the peer side. Furthermore, APEX helps peers to personalize navigation, prefetching and membership management, aiming at improving QoE for DVD-like functions by reducing response latency and optimizing content sharing. Both theoretical analysis and comprehensive simulations show that APEX outperforms most existing schemes in terms of accumulated hit ratio, response latency, and searching efficiency.

I. Introduction

The last few years have witnessed an explosive growth of Video-on-Demand (VoD) applications that provide users with free video access over the Internet. The proliferation of VoD-based Internet applications (e.g., Internet TV, online movie, distance education, and teleconferencing) attracts millions of Internet users while generating a large percentage of Internet traffic. Recently, Peer-to-Peer (P2P) technology has been proved to be an effective and resilient approach for "play-as-download" VoD applications in dynamic and heterogeneous networks [1], [2], [16], [18]. In P2P VoD applications, peers actively cache video contents and further relay them to other peers that are expecting these contents. Compared to traditional server-based solutions, P2P VoD applications can effectively alleviate server workload, save server bandwidth, and thus bring high system resilience and scalability.

A main challenge in P2P VoD applications is to support DVD-like functions, which currently has attracted great research interests [6], [13], [21], [23]. DVD-like functions include not only the VCR-like operations (e.g., random

seek, pause, fast forward, and rewind) but also some newly proposed functions (e.g., bookmark [5] and guided seek [14]) that help users to directly jump to the desired contents of the video. The free controls provided by DVD-like functions lead to frequent user interactivity in playback time, in a departure from the classic start-to-finish playback style. A lot of previous P2P VoD schemes try to support DVD-like functions via optimizing the P2P overlay to realize fast content searching [6], [21], [23]. In these schemes, when receiving DVD-like controls, the peers first search the requested video contents, then download them and finally playback. Such *locate-and-download* process leads to long response latency, which severely deteriorates users' Quality of Experience (QoE), e.g., playback freezing, even blackout. Thus, fast searching is necessary but far less sufficient for VoD applications.

Some existing works study how to improve QoE for DVD-like functions by modeling user interactive behavior, based on which predictive prefetching is used [13], [14], [24], [30]. Ideally, the video contents to be requested by future DVD-like functions are prefetched proactively so that users can always achieve zero response latency, i.e., view any portions of the video at anytime without any interruption. However, these works assume that all the users in P2P VoD applications have the same preference. As a result, all the peers' prefetching is directed by a same user behavior model reflecting the whole group preference. Different from these schemes, we improve QoE for DVD-like functions by utilizing *personalization* which accommodates the differences of preferences between users. Typically, different users have different preferences on video contents. With personalization, peers can perform recommendation and prefetching according to users' personal preferences. Moreover, personalization can help users to find their like-minded peers and thus optimize content sharing. In this paper, we present a novel video model that introduces *state* and *topic*, two powerful concepts that model both semantics of video contents and users' preferences over these contents. Our objective is to discover users' preferences so as to serve users proactively with their preferred contents and to make it easy for collaboration between users with close interests.

One simple solution for personalization is to let the server maintain all personalization information for each user. However, this method suffers from high bandwidth cost with low scalability. A distributed alternative for personalization is to let peers perform online mining based on the user logs collected through gossips. However, peers can hardly collect enough user logs in a short period, not to mention the heavy online mining cost. In this paper, we propose APEX, a hybrid personalization framework for P2P VoD applications, which leverages offline pattern mining on the server side and online personalization on the peer side. Since most P2P VoD systems employ certain bootstap servers (named *trackers*) to maintain large volumes of user logs [6], [18], [24], [25], APEX makes the tracker to mine *user access patterns*, which requires high computing cost and long computing time. Each peer receives these patterns at bootstrapping, and then achieves lightweight personalization taking both its active session and the patterns into consideration.

APEX peers achieve personalization through *Collaborative Filtering* (CF). CF aims at offering two main functions: find out the active user's preference, and look for who shares the same preference with the active user. With the help of CF, APEX improves QoE of DVD-like functions in three aspects: (1) Recommend users the video contents they may prefer (called *navigation*); (2) Prefetch contents that are likely to be requested by future DVD-like functions, aiming at achieving zero response latency; (3) Optimize membership management by clustering users with close

interests to improve the efficiency of content sharing.

Our contributions can be summarized as follows:

1) We present a novel video model that introduces *state* and *topic* to model both semantics of video contents and users' preferences over these contents. Based on the model, personalization can be effectively supported.
2) We propose a personalization framework for P2P VoD applications. The framework is highly practical to be integrated into most deployed P2P VoD systems. To the best of our knowledge, this is the first work taking account of personalization in P2P VoD applications.
3) We design personalized navigation, prefetching and membership management schemes in the personalization framework to improve QoE of DVD-like functions.

The rest of the paper is organized as follows. Sec. II reviews the related works. Sec. III presents the video model and system model. Sec. IV introduces the framework of APEX. In Sec. V, we design the personalized navigation, prefetching, and membership management. The QoE improvement for DVD-like functions is analyzed in Sec. VI. Sec. VII evaluates the system performance. Sec. VIII concludes the paper.

II. Related Work

Nowadays, using P2P technology to deploy VoD applications attracts great research interests [16], [18], [21], [23], [26]. Specially, there have been several works on supporting DVD-like functions by modeling and utilizing user interactive behavior in P2P VoD applications [13], [14], [17], [24], [30].

Huang and Hsu [17] propose a user-aware prefetching scheme which adopts simple Apriori algorithm to mine association rules from user logs. Prefetching is directed by these association rules. This scheme is designed for centralized web proxies and the simple Apriori algorithm is not efficient. Zheng et al. [30] propose a hierarchical prefetching scheme based on the optimal quantization theory. The scheme minimizes seeking distortion while gets high utilization ratio. VOVO [13] leverages all the modeling task on the peer side. Peers exchange user logs through gossips. By the collected user logs, a peer predicts its future behavior using on-line association rule mining, based on which hybrid prefetching is performed. He et al. [14] use the hybrid sketches to aggregate seeking statistics. Based on the statistics, an optimal prefetching scheme as well as an optimal cache replacement policy is developed to minimize the expected seeking delay. PREP [24] takes advantage of the tracker in the gossip-based P2P VoD systems to employ reinforcement learning to model user interactive behavior from large volumes of user logs. Each peer gets the model at bootstrapping and uses it to do predictive prefetching.

However, all these works assume that all the users in P2P VoD systems have the same preference, which is unreasonable in reality and cannot accommodate the preference differences between users. Different from these works, APEX is able to discover each user's preference, based on which peers can personalize navigation, prefetching and membership management to improve QoE of DVD-like functions.

III. Models and Assumptions

A. Video Model

In VoD applications, a video is divided into small segments of uniform length and each segment is equivalent to one unit time [23], [25], [29], i.e., segment is the basic unit for scheduling and transmission. However, the equal partition of segments contains no semantic information. Moreover, a video usually contains too many small segments, which makes it inefficient for predictive prefetching and navigation on the segment level. For example, a user may just want to jump to the next goal or the next penalty in the video but not the accurate segment like the 10-th segment. In this paper, we introduce *state* and *topic* to overcome these problems.

State models the association between video contents. A state consists of several continuous segments that are strongly associated, e.g., the segments that build up a goal can be considered as a state. Let $\mathcal{S} = \{s_i\}$ denote the set of all the states of a video, i.e., a video can be represented by $\mathcal{S}$. In the APEX framework, state is the unit for navigation and prefetching.

Topic models the semantics of video contents. In our video model, each video can be viewed as a finite mixture over an underlying set of topics, e.g., the topic of Goal and the topic of Penalty. Let $\mathcal{T} = \{t_j\}$ denote the set of all the topics in the video, where t_j is the j-th topic of the video.

Since a video is represented by states, each state s_i can be seen as a mixture over the topic set $\mathcal{T}$. Let $\phi_i = (\phi_{i1}, \cdots, \phi_{i|\mathcal{T}|})$ denote the probability distribution that state s_i is associated with topics in $\mathcal{T}$, i.e., ϕ_{ij} indicates the level of association between state s_i and topic t_j. We use the matrix $[\phi_{ij}]_{|\mathcal{S}|\times|\mathcal{T}|}$, called *state-topic matrix* to record the level of association between each state in $\mathcal{S}$ and each topic in $\mathcal{T}$.

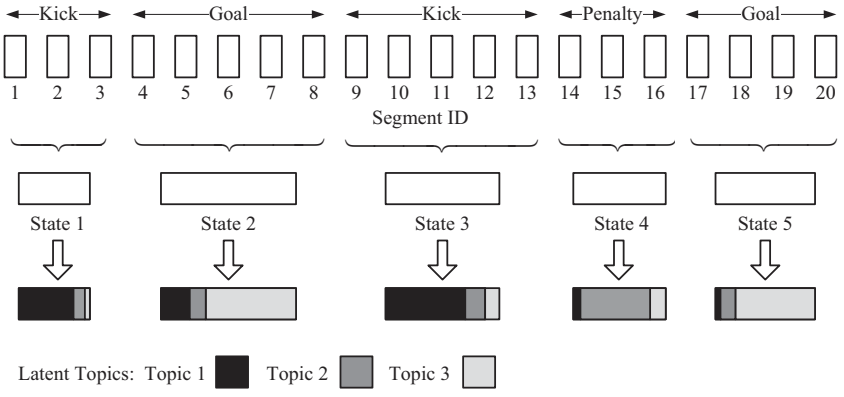

Fig. 1. Video model: segments, states and topics.

Fig. 1 demonstrates the relationship among segments, states, and topics in our video model. In Fig. 1, a video's 20 segments are partitioned into 5 states. Each state contains several strongly associated segments that build up a clip. There are 3 topics associated with the 5 states. Each state has a probability distribution over the 3 topics, which reflects the state's semantics (Kick, Goal, Penalty).

The state and topic also help to model users' preferences over video contents. A user session $\overrightarrow{u_k}$ can be represented by a *weighted state sequence* $\overrightarrow{u_k} = (w_1, \cdots, w_{|\mathcal{S}|})$, where w_i is the weight of state s_i in session $\overrightarrow{u_k}$. The weights can be binary, representing the existence or non-existence of the state in the session, or they may be a function of the occurrence or duration of the state in that session. A user session $\overrightarrow{u_k}$ also has a probability distribution over $\mathcal{T}$, denoted as $\theta_k = (\theta_{k1}, \cdots, \theta_{k|\mathcal{T}|})$, where θ_{kj} implies the level of association between user session $\overrightarrow{u_k}$ and topic t_j. θ_k reflects the topic preference (or interest) of the user who generates session $\overrightarrow{u_k}$. For example, users who like watching goals would generate sessions that are strongly associated with the topic of Goal. We use *session-topic matrix* $[\theta_{kj}]_{|\mathcal{U}|\times|\mathcal{T}|}$ to record the level of association between sessions and topics, where $\mathcal{U} = \{\overrightarrow{u_k}\}$ denotes the set of user sessions on the tracker.

B. P2P VoD System Model

A typical P2P VoD system mainly consists of three components: *media server*, *tracker* and *peer*, as illustrated in Fig. 2.

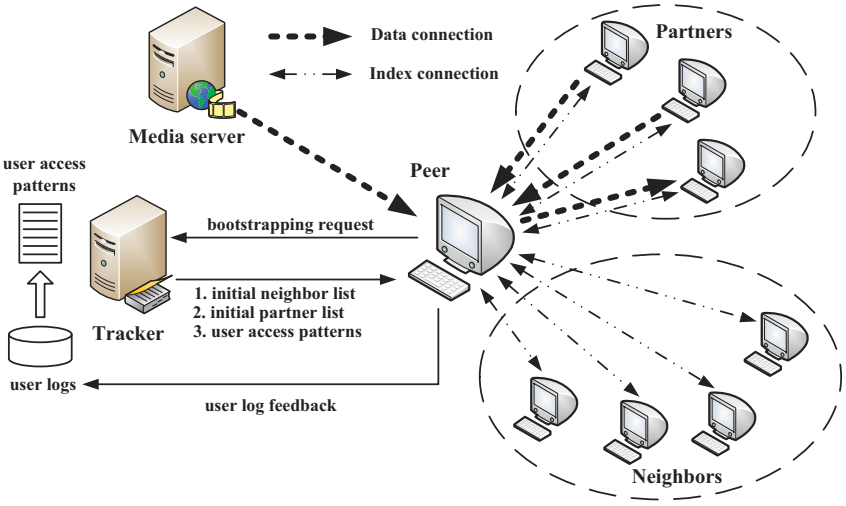

Fig. 2. P2P VoD system model.

The media server simply provides video streaming service. All video streams are initiated from the media server, delivered through the distributed P2P overlay, and finally reach end users. We assume that DVD-like functions are not supported by the media server, but by the coordination of peers. The tracker keeps track of users' playback and helps peers to connect to other peers at bootstrapping [7], [18], [25]. Thus, the tracker normally maintains large volumes of user logs.

Peers usually maintains two overlay networks: an *index overlay* for searching expected contents and a *data overlay* for exchanging contents. Since APEX's design aims to be compatible to most P2P VoD systems, we do not specify any index overlay (e.g., DHT [26], skip list [21]) and data overlay (such as derivative tree [23] and mesh [25]). Instead, we model the two overlays as follows: For the index overlay, each peer maintains a *neighbor pool*

TABLE I
THE USER VIEWING RECORD (UVR) FORMAT

User ID	Video ID	Start pos.	Duration	Jump Pos.

containing n neighbors by exchanging gossip messages through UDP; For the data overlay, each peer maintains a *partner list* containing m TCP-connected partners for content retrieving.

A peer retrieves two kinds of segments from partners: the urgent segments into a *playback buffer* for continuous playback and the predicted segments into a *prefetch buffer* to assist upcoming DVD-like functions. Both buffers are maintained as size-limited sliding windows to cache the most recently received segments, which could be supplied to other peers. We use B_l and B_r to denote the size of the playback buffer and the prefetch buffer.

IV. USER BEHAVIOR ANALYSIS

In this section, we first summarize some useful characteristics and analysis of user interactive behavior in VoD systems which inspire our design. Then we describe the user logs collected by the tracker.

A. Observation of User Behavior

The design of APEX is based on the following observations/conclusions investigated in many recent measurement studies [5], [7], [10], [12], [16], [27], [30].

1) VoD users often view only a portion of the full video stream, in a departure from the classic start-to-finish playback style. This phenomenon is more prominent for long videos (e.g., longer than 30 minutes [16]) such as movies and sports videos. Thus, peers in P2P VoD systems are highly dynamic and much difficult to collaborate with unstable connections.
2) There exist user access patterns to different period of a video stream. Some popular periods (e.g., hot scenes in movies, goals or penalties in sports videos) always attract far more user accesses than other non-popular periods. Such popularity is often described by skewed distributions such as Zipf-like distribution [27], and log-normal distribution [5].
3) A user's seeking position is not always accurate as expectation. Although new VCR-like features like *bookmark* [5] are proposed to help reduce browsing-like behavior, it is still hard for users to jump accurately. As a result, some existing VoD systems provide near segments (e.g., *anchors* [7], beginning segments [12]) instead of the accurate jump segment to satisfy user interactive requests.

B. User Log Collection

Since the most basic user activity is the continuous viewing of a stretch of a video, a APEX peer maintains such basic activity in a *user viewing record* (UVR) in playback time. The important parts of an UVR format is shown in Table I. In most cases, as soon as user finishes recording one UVR, a new UVR is initialized to

record the next viewing duration. A sequence of UVRs forms a *user viewing log* which represents a complete user viewing history. For example, $\{(U1, V1, 1, 6, 73), (U1, V1, 73, 3, 4), (U1, V1, 4, 3, \text{End})\}$ depicts a *viewing history* as follows: a user $U1$ first views the video $V1$ from the 1st second for 6 seconds, then seeks forward to the 73-th second and views until the 75-th second, and finally seeks backward to the 4-th second, re-views for 3 seconds and finishes viewing. User viewing logs can also be represented in *sequence format* as $\{s_0, s_1, \cdots, s_i, \cdots, s_{n-1}\}$, where s_i denotes that the user has viewed the s_i-th second of the video. In this example, the corresponding sequence format is $\{1, 2, 3, 4, 5, 6, 73, 74, 75, 4, 5, 6\}$.

V. The Personalization Framework

APEX is a hybrid personalization framework. Personalization is realized by the cooperation of the tracker and peers: APEX allows the peers collaboratively exchange topic and state information while utilizing minimal costly functions in the tracker. On one hand, we enable the already deployed tracker to mine topic-oriented user access patterns, which requires high computing cost as well as long computing time. On the other hand, after receiving the mined patterns at bootstrapping, each peer periodically performs lightweight collaborative filtering to find the active user's preferred topics and states based on the active session and the mined patterns. Note that the two parts are loosely coupled. The methods in each part can be customized depends on different conditions.

A. Mining User Access Patterns

The tracker mines the topic-oriented user access patterns in three steps: (1) Generate the state set $\mathcal{S}$ by extracting strongly associated segments into one state; (2) Compute the topic distribution of the state set $\mathcal{S}$ and session set $\mathcal{U}$, i.e., the state-topic matrix $[\phi_{ij}]_{|\mathcal{S}|\times|\mathcal{T}|}$ and the session-topic matrix $[\theta_{kj}]_{|\mathcal{U}|\times|\mathcal{T}|}$; (3) Construct the topic-oriented user access patterns $\mathcal{P} = \{\overrightarrow{p_1}, \cdots, \overrightarrow{p_{|\mathcal{T}|}}\}$, where $\overrightarrow{p_j}$ is the user access pattern corresponding to topic t_j.

Generate state set $\mathcal{S}$. The main idea of generating $\mathcal{S}$ is to partition the video stream according to the association between segments instead of uniform partition. The details of state partition can be found in our previous paper [24]. Having the state set $\mathcal{S}$, the tracker then map raw user sessions into weighted state sequence $\overrightarrow{u_k} = (w_1, \cdots, w_{|\mathcal{S}|})$. Taking all the sessions in $\mathcal{U}$ into consideration, the tracker maintains a $|\mathcal{U}| \times |\mathcal{S}|$ weight matrix $US = [w_{ki}]_{|\mathcal{U}|\times|\mathcal{S}|}$, where w_{ki} is the weight w_i in $\overrightarrow{u_k}$.

Compute topic distribution. Taking the weighted matrix US as input, the tracker computes the topic distribution of $\mathcal{S}$ and $\mathcal{U}$, i.e., computes the state-topic matrix $[\phi_{ij}]_{|\mathcal{S}|\times|\mathcal{T}|}$ and the session-topic matrix $[\theta_{kj}]_{|\mathcal{U}|\times|\mathcal{T}|}$. There are a number of topic models and techniques that are able to compute the two matrices revealing the relationship between topics, states and user sessions based on different assumptions, e.g., the LSA model [11], the PLSA model [15], the LDA model [4], etc. In the simulations, we use the LDA model as a representative.

Construct user access pattern. Based on the session-topic matrix $[\theta_{kj}]_{|\mathcal{U}|\times|\mathcal{T}|}$, the topic-oriented user access patterns $\mathcal{P} = \{\overrightarrow{p_1}, \cdots, \overrightarrow{p_{|\mathcal{T}|}}\}$ can be built. Defining a threshold μ, the tracker chooses user sessions that are strongly

associated with each topic t_j, i.e., user session $\overrightarrow{u_k}$ with θ_{kj} exceeds the threshold. We aggregate these chosen sessions to construct topic-oriented user access patterns. For each topic t_j, the topic-oriented user access pattern $\overrightarrow{p_j}$ can be built as: $\overrightarrow{p_j} = \sum \theta_{kj} \bullet \overrightarrow{u_k}$, subject to $\theta_{kj} > \mu$. By constructing user access pattern for each topic, we get the user access pattern set $\mathcal{P} = \{\overrightarrow{p_1}, \overrightarrow{p_2}, ..., \overrightarrow{p_{|\mathcal{T}|}}\}$.

B. Lightweight Collaborative Filtering

When launching P2P VoD applications, a peer usually makes DNS queries to get the tracker's IP address and then contacts the tracker for bootstrapping. During bootstrapping, an APEX peer receives the user access pattern $\mathcal{P}$, the state set $\mathcal{S}$ and the topic-state matrix $[\phi_{jk}]_{|\mathcal{T}|\times|\mathcal{S}|}$ from the tracker. Note that the size of the three configuration files are well bounded and would not cause high transmission overhead.

When playing the video stream, the peer periodically mines the preferred topics and states of the active user through collaborative filtering. The collaborative filtering is based on a similarity measure $Sim(\overrightarrow{u_c}, \overrightarrow{p_j})$ that measures the similarity between the active user session $\overrightarrow{u_c}$ and every mined pattern in $\mathcal{P}$. Here the active session $\overrightarrow{u_c}$ is also represented by weight state sequence $\overrightarrow{u_c} = (w_1^c, \cdots, w_{|\mathcal{S}|}^c)$ according to $\mathcal{S}$, and $w_i^c = 0$ if the user does not watch state s_i. There are a variety of similarity measures, including the Pearson's r correlation coefficient, the cosine coefficient, or the posterior probability $P(t_j|\overrightarrow{u_c})$. We use the cosine coefficient as a representative in our simulations.

Discover Preferred Topics (SAT-Set). Based on the measured similarities, the APEX peer builds the *Strongly Associated Topic Set* (SAT-Set) $\mathcal{O} = \{t_j | Sim(\overrightarrow{u_c}, \overrightarrow{p_j}) > \tau\}$ (τ is a pre-defined threshold). The SAT-Set $\mathcal{O}$ tells which topics the active user currently prefers.

Discover Preferred States (TAS-Set). To find which states the active user prefers, the peer calculates *Recommendation Score* R_i for each state s_i based on $\mathcal{O}$ and $[\phi_{ij}]_{|\mathcal{S}|\times|\mathcal{T}|}$: $R_i = \sum_{t_j \in \mathcal{O}} (Sim(\overrightarrow{u_c}, \overrightarrow{p_j}) \times \phi_{ij})$. Set $R_i = 0$ if the active user has already viewed state s_i. By sorting all the recommendation score in a descending order, the peer choose N states with top-N highest recommendation scores to build the *Top-N Associated State Set* (TAS-Set), which can be considered as the user's most preferred states.

Based on the SAT-Set and TAS-Set, the collaborative filtering offers two main functions: (1) Each peer can find out the active user's preferred topics and states, i.e., the user's *preference*, which helps the peer to personalize navigation and prefetching; (2) The peer can look for like-minded others who share similar preferences with the active user, which enlightens the peer to personalize membership management.

Note that the collaborative filtering is a lightweight online process. The computational complexity of finding SAT-Set and TAS-Set is well bounded by $O(|\mathcal{T}|)$ and $O(|\mathcal{O}| \times |\mathcal{S}|)$. In most cases, $|\mathcal{O}| < |\mathcal{T}| < 10$ and $|\mathcal{S}| < 500$.

VI. Personalizing Navigation, Prefetching and Membership Management

In this section, we present how APEX peers personalize their navigation, prefetching and membership management schemes according to the users' preferences.

A. Personalize Navigation

Personalizing navigation is straightforward, i.e., each peer shows the *navigation screenshots* of the states in TAS-Set to the user through the navigation interface. When a peer downloads segments in a state, it automatically generates the screenshot of that state. The screenshots are small in size compared with the video contents and can be stored as $cookies$ within the navigation interface. With the personalized membership (discussed in Sec. V-C), peers are likely to get a state's screenshot from partners and neighbors with similar topic preferences, not to mention that the media server is able to provide all these small shots directly.

B. Personalize Prefetching

For personalized prefetching, the peer tries to download the state with highest recommendation score in TAS-Set into its prefetch buffer. In APEX, peers prefetch the *anchors* [7], [12] of the selected state to improve the prefetching utilization ratio. When a jump is requested by a DVD-like function, the peer adjusts the playback position to the anchor of the requested state if the anchor has been already downloaded. It is reasonable to prefetch anchors because of the strong association among segments within each state.

APEX peers use *buffer map* (BM) to represent segment availability in their buffers. A peer periodically exchanges BM with its partners, and then schedules which segment is to be downloaded from which partner. The exchange period is called a *scheduling period*. APEX adopts a two-stage scheduling combining *fetching* and *prefetching*. For every scheduling period, a peer first utilizes the download bandwidth to fetch *urgent segments* for guaranteeing the continuity of normal playback. If there still residual download bandwidth left, the peer then performs prefetching. The urgent segment refers to the segment whose playback deadline is immediate. In APEX, we use the Urgent Line mechanism [19] in playback buffer, i.e., we have

$$id_{urgent} = \{\ id_x \mid id_{play} < id_x \leq (id_{play} + \alpha \times B_l)$$

where id_{urgent} is the ID of urgent segments, id_{play} is the ID of the current playing segment, α is the adaptive position of the urgent line. Note that the aggregate bandwidth usually exceeds the current user demand bandwidth in all within a significant period [16]. Thus, there is typically surplus upload capacity beyond what is needed to satisfy the normal playback demand. APEX uses a greedy rarest-first strategy trying to get the rarest segments as early as possible[1]. Segment i's rarity is considered as the maximum of the probability that it will remain in each partner's buffer, which we think is more reasonable than the traditional computation $rarity_i = 1/n'$, where n' is the number of peers that can supply segment i. The segment rarity in APEX is calculated through the following equation.

$$rarity_i = max\{\mathcal{R}_{i_1}, \mathcal{R}_{i_2}, \cdots, \mathcal{R}_{i_m}\}$$

[1]The problem that how to choose a proper supplier for every segment so that the number of missing segments is minimal is NP-hard (known as the parallel machine scheduling problem [9]).

$$\mathcal{R}_{i_x} = \begin{cases} 0 & i \text{ is not in peer } x\text{'s buffer} \\ p_l^{(i,x)}/B_l & i \text{ is in peer } x\text{'s playback buffer} \\ p_r^{(i,x)}/B_r & i \text{ is in peer } x\text{'s prefetch buffer} \end{cases}$$

where $p_l^{(i,x)}$ and $p_r^{(i,x)}$ denotes the segment i's position in the x-th partner's playback buffer and prefetch buffer respectively. Both the playback and prefetch buffer use the LRU policy as the buffer replacement strategy [22].

C. Personalize Membership Management

In APEX, peers personalize their membership management by exploiting the social networks' Friend-to-Friend (F2F) relation [8]. Peers with similar preferences can be seen as like-minded friends and are more likely to share their buffered contents. The basic idea is to organize peers into different *Topic Clusters* (TC). Each TC corresponds to one topic in $\mathcal{T}$. A TC_i is made up of peers interested in topic t_i. All the TCs compose the upper level overlay (called TC-based overlay). Obviously, an APEX peer may appear in multiple TCs because of the multiple topics in its current SAT-Set. Both the media server and tracker are by default in every TC to guarantee playback continuity and interconnection. Fig. 3 offers an example, where there are 3 TCs for topic t_1, t_2 and t_3 respectively in the TC-based overlay. Peer v_1 is interested in all the three topics and thus appears in the three TCs. As a result, the three TCs have mutual neighboring relations. Once peer v_1 updates its preference and takes no interest in topic t_2, it leaves TC_2. In this case, the mutual neighbor relation still exists because of the common peers: v_1 and v_2.

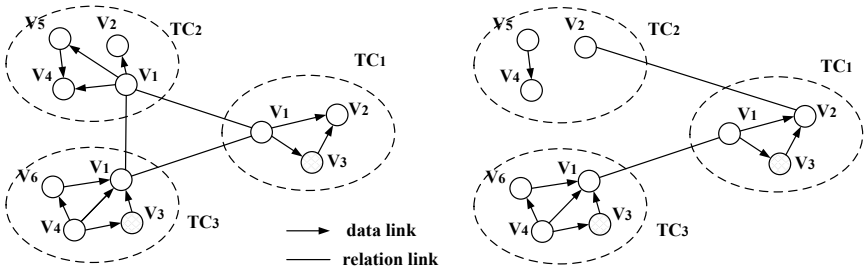

Fig. 3. Illustration of a TC-based overlay.

The TC-based overlay is made up of conceptual relations by partner/neighbor selection, i.e., peers are inclined to select their friends as their partners and neighbors. When a new peer enters the P2P VoD application, the tracker, keeping track of the TCs, redirects the peer to the corresponding TCs. When streaming the video, the peer computes the SAT-Set $\mathcal{O}$ in each scheduling period and distributes $\mathcal{O}$ through gossip messages. Each peer continually overhears the gossip messages passing by, and updates both the partner list and neighbor pool giving peers with similar preferences the higher priority. In this way, the TC-based overlay comes into being.

The TC-based overlay significantly improves the efficiency of content sharing while reduces the searching overhead because of the F2F relations. When searching segments due to DVD-like functions, the peers will be

very likely to find the expected segments in their friends' buffers because of their close interests. In the worse case that a peer cannot find a segment through its partners or neighbors, the peer searches state s_i that contains the segment in TC_k, where k satisfies that state s_i is strongly associated with topic t_k. Suppose that the number of states associated with topic t_k is z_k, the user preferences within each topic follows the uniform distribution, and each peer in TC_k contains n_k of z_k states. Then, the probability that the peer can find the required states in TC_k satisfies:

$$P(\text{Search success in TC}_k) \geq 1 - (1 - \frac{n_k}{z_k})^{|\mathcal{C}_k|} \approx 1 - e^{\frac{|\mathcal{C}_k|}{\frac{z_k}{n_k}}}$$

where $\mathcal{C}_k$ denotes the set of all the peers in TC_k. The personalized membership management makes it easy for peers to discover expected segments from friends, while improves the searching efficiency by reducing the searching space.

VII. QoE Improvement of DVD-like Functions

Most DVD-like functions can be implemented by the jump process: the combination of leaving the current playback position and requesting a new position. A guided seek, bookmarking or pause jumps only once, and a fast forward/backward generally consists of a series of jump process [21]. Moreover, as reported in [10], [12], fast forward/backward operations occupy little (about 1%) in the workload while the majority are seeks (48%) and pauses (51%). Thus, we focus on the jump process in this section.

In current P2P VoD applications, once a peer requests a DVD-like function, the peer should first locate the requested segment and then download it for playback. This *locate-and-download* process leads to playback freezing due to long *response latency* (denoted as Lat). In this case,

$$Lat = T_{search} + T_{conn} + T_{down}$$

where T_{search} is the searching latency, T_{conn} is the time for building streaming connection and T_{down} denotes the time for downloading the requested segments.

APEX effectively reduces the response latency through prefetching and efficient searching. An APEX peer handles a jump request according to the following 4 cases:

1) If the requested segment is already prefetched by the current peer, the peer directly plays out the segment without any sojourn time. In this case,
$$Lat_1 = 0$$

2) If the requested segment is not cached on the current peer but downloaded on its partners' buffer, the peer downloads the segment from the partner and resumes playback. In this case,
$$Lat_2 = T_{down}$$

3) If the requested segment is neither cached on the current peer nor on partners but downloaded by its

neighbors, the peer first builds streaming connection with the neighbor, then downloads the segment and resumes playback. In this case,

$$Lat_3 = T_{conn} + T_{down}$$

4) If the requested segment is not cached on the local buffer nor cached by the partners or neighbors, the peer has to launch the locate-and-download process. In this cases,

$$Lat_4 = T_{search} + T_{conn} + T_{down}$$

Suppose the occurring possibility of the Case 1 – 4 above is P_1, P_2, P_3, and P_4 respectively ($P_1+P_2+P_3+P_4 = 1$). Then the expectation of response latency is,

$$E[Lat] = P_1 \times E[Lat_1] + P_2 \times E[Lat_2] + P_3 \times E[Lat_3] + P_4 \times E[Lat_4]$$

It is clear that:

$$E[Lat_1] < E[Lat_2] < E[Lat_3] \ll E[Lat_4]$$

Thus, higher P_1, P_2 and P_3 leads to lower $E[Lat]$. Note that P_1 is decided by the *hit ratio* of prefetching. A higher accuracy of prefetching implies a higher P_1. The personalized navigation and prefetching of APEX could achieve high *hit ratio* and thus decrease the response latency. On the other hand, with the help of the TC-based overlay, peers can achieve high P_2 and P_3, especially with large user population.

VIII. Performance Evaluation

In this section, we evaluate the performance of APEX via simulations, and compare it with other solutions.

A. User Log Generation

In the simulations, we use the user logs collected from a real deployed VoD system [3] to generate user viewing history. The original user log set is composed of a large number of structures which contains users' IP addresses, inter-seek duration, and video information. We can find out how many users watched a same video according to the the unique program ID in these logs. We take a movie of 8000 seconds as a test video which has 4338 history logs of user sessions. Thus, we have enough user session history for mining and prediction. The average duration length is 44.60 seconds and average number of DVD-like functions within a session is 5.22, i.e., each user watches the video for about 232.86 seconds and perform about 5 DVD-like controls in average.

For the 8000-second video, we have already know that the video is a football video and is composed of "Goal", "Penalty", "Shoot" and "Card". Based on these facts, we let 50% users prefer "Goal", 30% users prefer "Penalty", 30% users prefer "Shoot", 10% users prefer "Card". Users trends to jump to their preferred video contents after finishing one inter-seek duration according to the user logs. Thus, we can get user history logs in UVR format.

For the 4338 user history logs, we use 1338 logs as training set to build the CF-based prefetch model (Sec. VII-C(1) shows that 1338 logs is enough for mining). We simulate 3000 peers by generating their playback sessions

according to the rest 3000 history logs. For example, a peer with the viewing history (0, 1, 2, 3, 6, 7, 8, 15, 16, 45, 30, 31, 32, 46, 47, 48) has a life time of 16 seconds and performs 5 DVD-like functions in total.

B. Simulation Methodology

We use packet-level event-driven simulations to evaluate APEX. The simulations are built on top of a topology of 4000 peer nodes based on the transit-stub model generated by GT-ITM [28]. The streaming rate is $S = 256Kbps$ (most video stream over the Internet today is encoded in the 200 – $400Kbps$ range [16]). The default size of the playback buffer and prefetch buffer, denoted as B_l and B_r, is $25Mbytes$ and $5Mbytes$ respectively, i.e., each peer can cache 100 recent segments and 20 predicted segments in total. The arrival of peers follows the Poisson Process with $\lambda = 5.4$ according to the hot-time arrival rate in [27]. After joining the system, a peer is allocated a playback session randomly from the history logs. When finishing playback, the user immediately disconnects and leaves the system. We set both the download bandwidth and the upload bandwidth of each peer be randomly distributed in $[256Kbps, 768Kbps]$ and let the average bandwidth be both 512 Kbps. Each peer can have 5 partners and 10 neighbors. The data scheduling period is 5 seconds. The average TCP connection latency is set as $200ms$. The physical latency between two overlay node (i.e., one overlay hop) is from $25ms$ to $100ms$ according to the physical distance in the topology. Moreover, we limit the end-to-end bandwidth to the streaming rate, i.e., the download latency for one segment is at least 1 second.

C. Implement Collaborative Filtering

We use the LDA model [4] as the topic model to compute the state-topic matrix $[\phi_{ij}]_{|\mathcal{S}|\times|\mathcal{T}|}$ and session-topic matrix $[\theta_{kj}]_{|\mathcal{U}|\times|\mathcal{T}|}$ in our simulations. LDA provides a probabilistic approach for the discovery of topics while suffers little from the overfitting problem. We use the GibbsLDA++ [20] as the LDA implementation, setting α = 5.0, β = 1.0, the default number of topics as 10, and the number of Gibbs sampling iterations as 2000.

We use the cosine coefficient as the similarity measure $Sim(\overrightarrow{u_c}, \overrightarrow{p_j})$ in our simulations, i.e, $Sim(\overrightarrow{u_c}, \overrightarrow{p_j}) = (\overrightarrow{u_c} \bullet \overrightarrow{p_j})/(\|\overrightarrow{u_c}\| \times \|\overrightarrow{p_j}\|)$, where $\bullet$ is the dot product of vectors and $\|\cdot\|$ is the vector norm. The cosine coefficient measures the cosine angle between two vectors.

D. Simulation Results

1) **Performance of CF-based Model:** We first evaluate APEX's CF-based prefetch model in terms of *hit ratio* in a simple prediction experiment. The hit ratio decides the performance of navigation and prefetching. In the simple experiment, each peer predicts the next jump position according to the CF-based model, and adds the corresponding segments into its prefetch buffer at each playback slot. When a jump occurs, if the prefetch buffer contains segments in the jumping state, we count a hit event. Here we do not consider bandwidth constraints, peer collaboration and other network conditions. Notice that the experiment is like in a C/S VoD application, i.e., each peer well connects to a powerful media server. We compare APEX's CF-based model with the prefetch model in VOVO [13], statistic-based optimal prefetching scheme (labeled as SOPS) [14] and PREP [24]. Fig. 4 shows the average hit ratio of

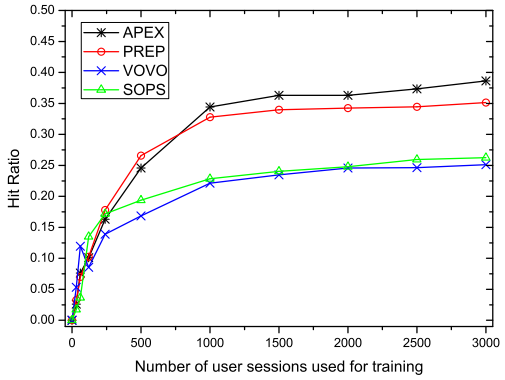

Fig. 4. Average hit ratio in simple prediction.

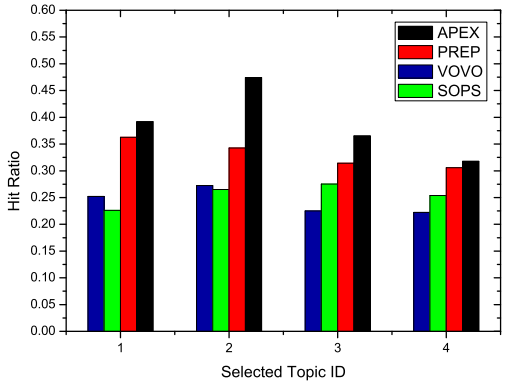

Fig. 5. Average hit ratio of different topics.

the 4 prefetch model in the simple prediction experiment using different size of training set. We can see that the CF-based model in APEX has higher hit ratio than the others. When the number of user sessions for training is about 1500, the average hit ratio exceeds 35%. Considering the large state space (the test movie is divided into 419 states), APEX has high performance for the multi-class prediction. Moreover, when the training set size exceeds 1000, the hit ratio trends to stable. Increasing the training set size cannot improve the hit ratio significantly. As a result, we choose to use 1338 history logs to build the prefetch model.

Fig. 5 illustrates the average hit ratio of users with different topic preferences in the simple prediction experiment. We show 4 typical topics from the 10 topics since most users has the 4 topic preferences. We can see that the CF-based model in APEX has higher hit ratio than the other prefetch models. In addition, the CF-based model has a special high prediction performance for topic 2 which contains about 50% of the whole user population.

2) **Performance of Collaborative Prefetching:** Due to limit of the prefetch buffer size, a peer cannot prefetch all the predicted segments into its buffer. This could be compensated by peer collaboration through gossip protocol. We evaluate the performance of APEX in terms of accumulated hit ratio in a typical gossip-based P2P VoD application. We let 3000 peers join the system with the arrival rate $\lambda = 5.4$ and record each peer's hit ratio. We differentiate the

hit ratio in the local buffer, the hit ratio in the partners' buffer, and the hit ratio in the neighbors' buffer, denoted as LHR, PHR, and NHR respectively. The three kinds of hit ratio are corresponding to Case 1 – 3 in Sec. VI. We further define Accumulated Hit Ratio (AHR) and Semi-Accumulated Hit Ratio (SHR): AHR = LHR + PHR + NHR, and SHR = LHR + PHR. Moreover, we use three prefetching strategies in our simulations, named Full-Server Prefetching (FSP), Semi-Server Prefetching (SSP), and No-Server Prefetching (NSP). In FSP, peers first try to prefetch predicted segments from partners. If a peer fail to prefetch these segments, it pulls the segments from the media server. In NSP, peers never pull predicted segments from the media server. In SSP, peers only pull the segments whose recommendation score R_i is larger than a threshold (0.1 in our simulations).

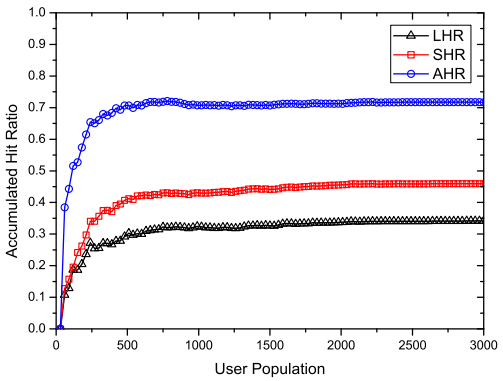

Fig. 6. Accumulated hit ratio using FSP.

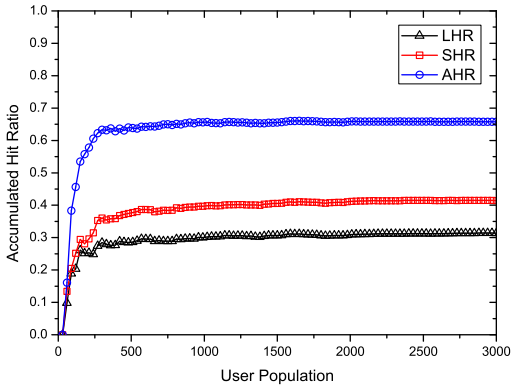

Fig. 7. Accumulated hit ratio using SSP.

Fig. 6 – 8 show the accumulated hit ratio using FSP, SSP, and NSP respectively. According to the three figures, when the user population exceeds 1000, the LHR, SHR and AHR trend to stable. When the user population is 3000, the LHR, SHR, AHR is 35%, 45%, 70% for FSP, 32%, 40%, 65% for SSP, and 25%, 35%, 53% for NSP. Obviously, FSP has the highest hit ratio. The LHR of FSP (35%) is close to the hit ratio in the simple prediction experiment (38%) when the user population is 3000. However, FSP imposes the highest server stress since some unnecessary segments would be pulled from the server (discussed in Sec. VII-B(5)). Since the SSP has approximate

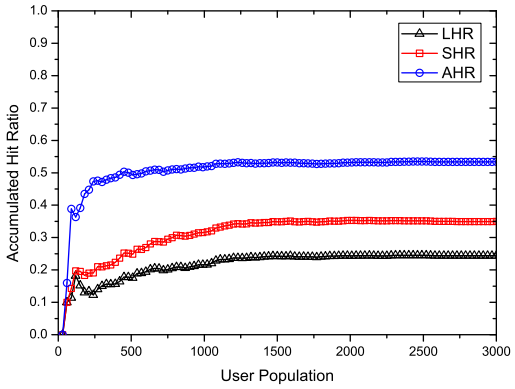

Fig. 8. Accumulated hit ratio using NSP.

collaboration prefetching performance as FSP while brings lower extra server stress, we choose SSP in the following simulations.

3) **Searching Efficiency:** We evaluate the searching efficiency using the TC-based overlay in terms of two metrics: the overlay hops for searching the first supplier and the number of searched suppliers in the maximum allowable delay. When a peer cannot satisfy a DVD-like request, it search the requested segments in the overlay. The less the hop is, the faster the VoD application can find the supplier and resume playback. By tracing the searching messages, we record average searching hops for locating the first supplier in APEX. Fig. 9 shows the average searching hops of APEX for different number of neighbors, denoted as n, compared with the system that does not adopt TC-based overlay (labeled as NO-TC). We can see that the TC-based overlay effectively reduces the searching hops to find an available supplier. When the number of average searching hops is stable, the NO-TC system uses about 3 hops to find an available supplier while the APEX uses only about 2.30 hops. Moreover, increasing the number of neighbors reduces the searching hops. From Fig. 9, the average searching hops for $n = 20$ is only about 1.75. On the other hand, large n indicates heavy maintenance overhead.

Fig. 10 shows the average number of searched suppliers in the maximum allowable delay (MAD). We set the MAD as $200ms$ in our simulations. The number of searched suppliers measures the ability of content sharing. According to Fig. 10, the TC-based overlay can find more suppliers than NO-TC overlay in the same MAD, even in the case of small n. Taking the advantage of topic clusters, APEX resumes high-quality playback quickly after DVD-like requests.

We would be remiss if not discuss the relation between searching efficiency and the number of TCs which equals to the topic number $|\mathcal{T}|$. $|\mathcal{T}|$ is mined by LDA and thus is not a tunable parameter. In most cases, $|\mathcal{T}| < 10$. In our simulations, we preset $|\mathcal{T}| = 10$ and find the top 4 topics occupy over 90% users in the mined result.

4) **Response Latency:** Low response latency means that the VoD system responds quickly to DVD-like functions, which leads to high user QoE. Remember that there are 4 cases of response latency discussed in Sec. VI. The CF-

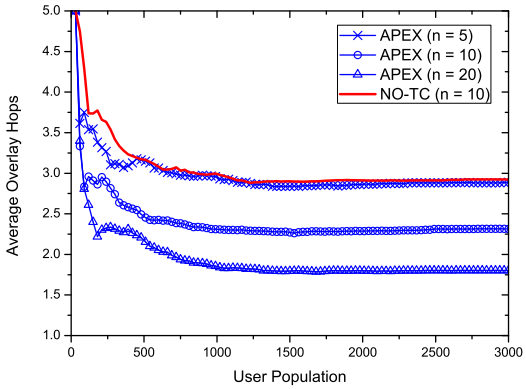

Fig. 9. Average searching hops using TC.

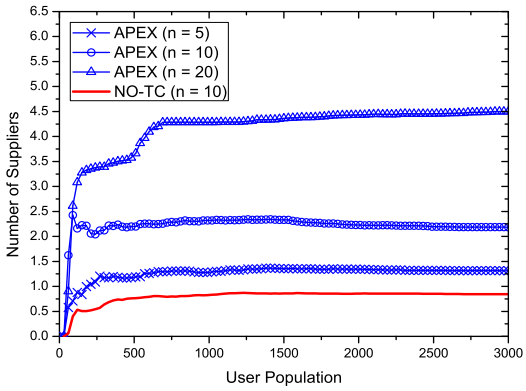

Fig. 10. Average searched suppliers using TC.

based prefetch model as well as peer collaboration tries to reduce Lat_1, Lat_2 and Lat_3, while the TC-based overlay tries to reduce Lat_4. We trace the response latency of APEX with different prefetch buffer sizes B_r and compare them with that of the system with neither TC-based cluster overlay nor prefetching (labeled as NO-TP). Fig. 11 shows average response latency per DVD-like request. We can see that APEX reduces the average response latency significantly by using prefetching and TC-based overlay. With the same prefetch buffer size (20Mbytes), the response latency of APEX is only about 60% of that of NO-TP, which means APEX achieves about 60% quicker response. We also discuss APEX's response latency with different prefetch buffer size B_r. As is shown in Fig. 11, when prefetch buffer size is small, increasing B_r makes response latency decrease quickly. For example, when B_r increases from 5Mbytes to 10Mbytes, the response latency drops $70ms$. However, while prefetch buffer is able to contain most recent predicted segments, increasing B_r will not improve response latency significantly. At this time, the response latency is mainly bounded by the prefetching performance and the searching efficiency.

5) **Prefetch Overhead:** In FSP and SSP, to prefetch a segment (with high confidence), an APEX peer first tries to pull the predicted segment from its partners. If no partner contains the predicted segment, the peer tries to find that segment from its neighbors. If the peer fails to prefetch the segment in the predefined allowable delay (1.5

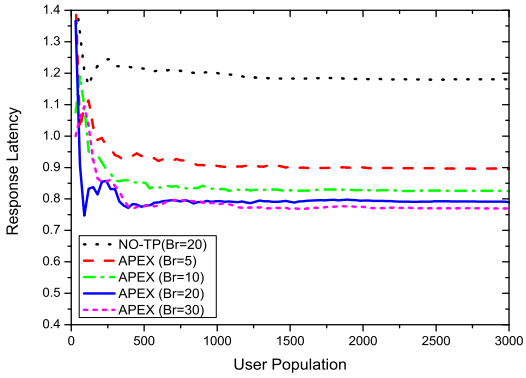

Fig. 11. Avg. response latency per DVD-like function.

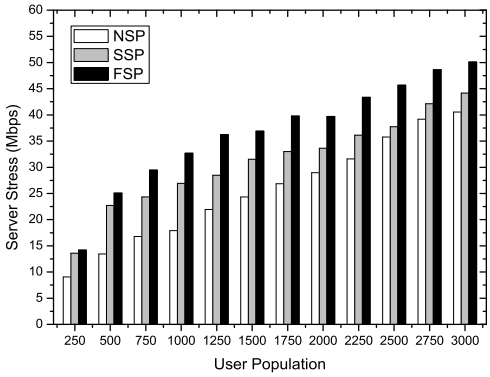

Fig. 12. Prefetching overhead.

second in our simulations), the peer pulls the segment directly from the media server, which causes extra server stress. Fig. 12 shows the server stress using FSP and SSP, compared with NSP. Remember that peers using NSP do not pull any predicted segments from the media server. As a result, NSP imposes no prefetch overhead on the server side and its server stress can be considered equal to that of NO-TP. We define O_f and O_s as the prefetch overhead for FSP and SSP: $O_f = \frac{S_f}{S_n}$, $O_s = \frac{S_s}{S_n}$, where S_f, S_s, and S_n is the server stress for FSP, SSP, and NSP respectively. From Fig. 12, we can see that prefetch overhead imposed by FSP and SSP decreased quickly with the increasing of user population. For example, when user population is 1000, O_s=1.505 and O_f=1.826. When user population is 3000, O_s=1.089 and O_f=1.235. This is because large user population brings high coverage of segments in the whole system. When the coverage of segments is dense enough, it is easier for peers to fetch the predicted segments from partners and neighbors rather than pulling them directly from the media server. Not surprisingly, SSP brings less overhead than FSP because peers using SSP do not prefetch low confident segments from the server and thus avoid pulling too many unnecessary segments.

IX. Conclusions and Future Work

In this paper, we propose APEX, a practical personalization framework in P2P VoD applications. To the best of our knowledge, APEX, for the first time, attempts to achieve personalization in P2P VoD applications, aiming at improving QoE for DVD-like functions. APEX is based on the presented video model that characterizes the personalization information for users' contents and preferences. Moreover, APEX uses a hybrid architecture that is practical and deployable for most P2P VoD systems. We show how APEX may help peers to personalize their navigation, prefetching, and membership management schemes. The effectiveness of APEX is verified through theoretical analysis and extensive simulations.

We plan to improve our work in the following aspects. First, as the first attempt for personalization in P2P VoD applications, a specific available dataset was used in our case study. It would be useful to study how APEX performs and could be improved if necessary based on further analysis of large-scale user traces and other datasets. Second, with the booming of YouTube-like sites, short video clip sharing becomes ever popular. In this scenario, the main challenge of improving QoE turns to shorten the startup delay [8]. We are interested in extending the idea of personalization to optimize P2P short video sharing. Third, we plan to further understand the APEX performance when the personalized membership management is integrated into specific index overlays (e.g., skip list [21], DHT [26], RINDY [6]).

Acknowledgement

We thank Yunhao Liu and Yuan He of Hong Kong University of Science and Technology for their valuable inputs and suggestions.

References

[1] "PPLive", http://www.pplive.com/, 2010.

[2] "Joost", http://www.joost.com/, 2010.

[3] "PowerInfo Co., LTD", http://www.sjdd.com.cn/english/englishindex.htm/, 2007.

[4] Y. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," In *Journal of Machine Learning Research*, 3:993-1022, 2003.

[5] A. Brampton, A. MacQuire, I. A. Rai, N. J. P. Race, L. Mathy, and M. Fry, "Characterising User Interactivity for Sports Video-on-Demand," In *Proc. of ACM NOSSDAV'07*, Urbana-Champaign, USA, Apr. 2007.

[6] B. Cheng, H. Jin, and X. Liao, "Supporting VCR Functions in P2P VoD Services Using Ring-Assisted Overlays," In *Proc. of IEEE ICC'07*, Glasgow, Scotland, Jun. 2007.

[7] B. Cheng, X. Liu, Z. Zhang, and H. Jin, "A Measurement Study of a Peer-to-Peer Video-on-Demand System," In *Proc. of IPTPS'07*, Belleue, USA, Apr. 2007.

[8] X. Cheng and J. Liu, "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing," In *Proc. of IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.

[9] T. Cormen, C. Leiserson, and R. Rivest, "Introduction to Algorithms," *MIT Press*, Cambridge, MA, USA, 1990.

[10] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto, "Analyzing Client Interactivity in Streaming Media," In *Proc. of ACM WWW'04*, New York, USA, May. 2004.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, and T. K. Landauer, "Index by Latent Semantic Analysis," *Journal of the American Society for Information Science*, 41(6):391-407, 1990.

[12] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "DISC: Dynamic Interleaved Segment Caching for Interactive Streaming," In *Proc. of IEEE ICDCS'05*, Columbus, USA, Jun. 2005.

[13] Y. He and Y. Liu, "VOVO: VCR-Oriented Video-On-Demand in Large-Scale Peer-to-Peer Networks," In *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 20(4):528-539, 2009.

[14] Y. He, G. Shen, Y. Xiong, and L. Guan, "Optimal Prefetching Scheme in P2P VoD Applications With Guided Seeks," *IEEE Transaction on Multimedia*, 11(1):138-151, 2009.

[15] T. Hofmann, "Probabilistic Latent Semantic Analysis," In *Proc. of UAI'99*, Stockholm, Sweden, Jul. 1999.

[16] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-Demand be Profitable?" In *Proc. of ACM SIGCOMM'07*, Kyoto, Japan, Aug. 2007.

[17] C. M. Huang and T. H. Hsu, "A User-Aware Prefetching Mechanism for Video Streaming," *World Wide Web: Internet and Web Information Systems*, Kluwer Academic Publishers, 6(4):353-374, 2003.

[18] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Liu, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," In *Proc. of ACM SIGCOMM'08*, Boston, USA, Aug. 2008.

[19] Z. Li, J. Cao, and G. Chen, "ContinuStreaming: Achieving High Playback Continuity of Gossip-based Peer-to-Peer Streaming," In *Proc. of IPDPS'08*, Miami, USA, Apr. 2008.

[20] X. H. Phan and C. T. Nguyen. "GibbsLDA++: A C/C++ Implementation of Latent Dirchlet Allocation (LDA) using Gibbs Sampling for Parameter Estimation and Inference," http://gibbslda.sourceforge.net/.

[21] D. Wang and J. Liu, "A Dynamic Skip List-based Overlay for On-Demand Media Streaming with VCR Interactions," In *IEEE Transaction on Parallel and Distributed Systems (TPDS)*, 19(4):503-514, Apr. 2008.

[22] J. Wu and B. Li, "Keep Cache Replacement Simple in Peer-Assisted VoD Systems," In *Proc. of IEEE INFOCOM'09 Mini-Conference*, Rio de Janeiro, Brazil, Apr. 2009.

[23] T. Xu, J. Chen, W. Li, S. Lu, Y. Guo, and M. Hamdi, "Supporting VCR-like Operations in Derivative Tree-Based P2P Streaming Systems," In *Proc. of IEEE ICC'09*, Dresden, Germany, Jun. 2009.

[24] T. Xu, W. Wang, B. Ye, W. Li, S. Lu, and Y. Gao, "Prediction-based Prefetching to Support VCR-like Operations in Gossip-based P2P VoD Systems," In *Proc. of IEEE ICPADS'09*, Shenzhen, China, Dec. 2009.

[25] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Rodriguez, "Kangaroo: Video Seeking in P2P Systems," In *Proc. of IPTPS'09*, Seattle, USA, Apr. 2009.

[26] W. P. K. Yiu, X, Jin, and S. H. G. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," In *IEEE Journal on Selected Areas in Communications*, 25(9):1717-1731, Dec. 2007.

[27] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-Scale Video-on-Demand Systems," In *Proc. of ACM EuroSys'06*, Leuven, Belgium, Apr. 2006.

[28] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," In *Proc. of IEEE INFOCOM'96*, San Francisco, USA, 1996.

[29] X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," In *Proc. of IEEE INFOCOM'05*, Miami, USA, Mar. 2005.

[30] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," In *Proc. of ACM P2PMMS'05*, Singapore, Nov. 2005.