SocialVoD: a Social Feature-based P2P System

Wei Chang and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122
Email: {wei.chang, jiewu}@temple.edu

***Abstract*—Video-on-demand (VoD) service has been explosively growing since its first appearance. For maintaining an acceptable buffering delay, the bandwidth costs have become a huge burden for the service providers. Complementing the conventional client-server architecture with a peer-to-peer system (P2P) can significantly reduce the central server's bandwidth demands. However, the previous works focus on establishing a P2P overlay for each video, producing a high maintenance cost on users. Per-channel-based overlay construction was first introduced by SocialTube [1], which clusters the users subscribed to the same video channels into one P2P overlay. However, the current per-channel overlay structure is not suitable for users developing new watching preferences. Consider that a channel's subscribers tend to watch not only the videos from the channel, but also other videos from similar channels. In this paper, we propose a new overlay structure by exploring the existing social relations of users and the similarities of video channels. Our system creates a hierarchical overlay: subscribers of the same channel form the low-level overlay (also known as groups), and in high-level overlay, different groups are connected based on their similarities. The new structure has the small-world property, the existence of which has been found in most data-sharing patterns. Based on the new structure, we propose a routing algorithm for both channel subscribed and unsubscribed users. Extensive simulation results show the efficiency of our approach.**

***Keywords*—Community, peer-to-peer (P2P) systems, social features, subscription, video-on-demand (VoD).**

I. Introduction

In the past decade, we have witnessed the enormous growth of video-on-demand (VoD) services, such as Vimeo and YouTube. According to sites [2]–[4], Vimeo attracted over 100 million unique visitors per month in 2013. With the development of affordable smartphones and Internet, more and more VoD platforms have allowed users to create their own media channel and upload videos. For example, Ray Johnson has the most popular personal Youtube channel [5], which has gained 10 million subscribers. Clearly, a tremendous amount of network resources have been used to maintain such a huge supply and demand market. Most of the current VoD systems are based on a client-server (CS) architecture. All videos are stored in central servers; whenever a user tries to watch a video, a query message will be generated and sent to the servers; then, the corresponding video will be downloaded from the servers to the requester. Although the CS architecture is easy to manage, it brings enormous bandwidth costs for the servers' provider, and long buffering delays for users.

In order to reduce the bandwidth costs and users' average delays, complementing the CS architecture with a peer-to-peer (P2P) architecture becomes a common approach. In a P2P architecture, users, also known as peers, are both video-resources' suppliers and consumers. Instead of fully depending on central servers, in P2P, peers are able to download videos from other peers, which significantly lightens the servers' burdens. Peers are logically connected at the protocol layer, which builds a network overlay on the top of the underlying physical network [6]. The research on P2P has existed for decades, and the main concern is how to construct such a logical overlay for efficiently locating the files. Representative systems for unstructured P2P include PA-VOD [7] and NetTube [8], and the typical structured P2P system is HyperCuP [9]. However, all of these systems are based on a per-video level: for each video, an overlay graph is constructed. As pointed out by paper [1], the per-video overlays not only generate prohibitive costs for maintaining the overlays, but also create plenty of redundant links between a pair of nodes on different overlays.

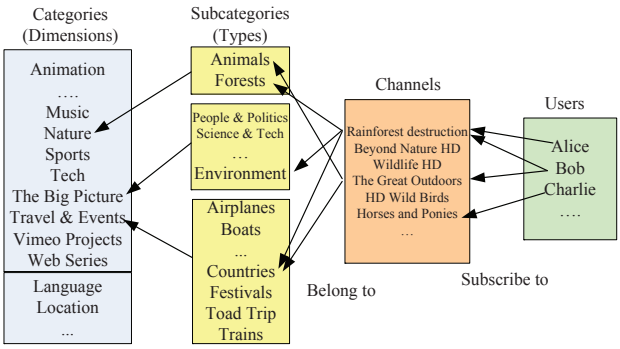


Fig. 1. Vimeo's feature space structure. In Vimeo, videos, channels, and groups are organized by categories. Unlike the conventional tags, categories are system-defined, and each category consists of several subcategories. Based on the content of video channels, the channel owners and Vimeo staff can associate several category values with the channels.

To resolve these problems of the conventional P2P systems, SocialTube [1] leverages users' actual subscription relationships. Many VoD sites provide channel subscription functions to users. A user can create his own media channel webpages, and each channel contains some videos related to its topic. For easily browsing channels, most VoD sites classify channels into categories, according to their features, and each category may have different types of values (i.e. subcategories). Fig. 1 gives an example of the category structure in Vimeo. If a registered user is interested in a channel, and wants to track all videos from the channel, he can subscribe to it and keep receiving update feeds whenever new videos are uploaded. Consider that the subscribers of a channel are likely to watch the same videos from the channel, and that users with similar interests tend to access similar videos. SocialTube proposes an interest-based per-channel hierarchical overlay: peers subscribed to the same channel are built into one low-level community, and the channels of each category are formed into a high-level community. When a video query is generated, random walk-based searching is conducted within these communities. The per-channel overlay structure can quickly locate the resources

when users search for their subscribed channels' videos. Compar to the traditional per-video overlay, the new structure has both low maintenance costs and high QoS.

SocialTube has two problems. First, it is not efficient for subscribers to explore videos belonging to other categories. Users' watching preferences (i.e. interests) are not a fixed value; one may find his new watching preferences by exploring other categories' channels. Take Fig. 1 as an example. The subscribers of "The Great Outdoors" channel are also likely to watch the videos from "Rainforest Destruction". By using SocialTube, one has to go through all the searching space of either 'Nature" or "Travel & Events" categories to find the target channel's videos. In addition, if the subscribers of "The Great Outdoors" want to watch videos in the "Animation" category, it is hard to locate the videos on SocialTube's overlay, since there is no efficient overlay connecting across different categories. Second, SocialTube is not applicable for the VoD sites (e.g. Vimeo), who are rich in category information. Note that most channels are naturally suitable to be labeled by multiple categories. For instance, "The Great Outdoors" belongs to both the "Animals" subcategory of the "Nature" category and "Countries" subcategory of "Travel & Events". As a result, the high-level communities, created by SocialTube, are actually overlapped with each other, and eventually form into one community, instead of several communities.

In this paper, we propose a new hierarchical P2P overlay structure, call SocialVoD, by exploring both the subscription relationships and channels' similarities on Vimeo. The key design of SocialVoD is the utilization of channel subscribers' social closeness: the closer the watching preferences of users are, the better the overlay connectivity should be between them. We not only build interest-based per-channel overlays, but also, in a high-level, we organize these overlays according to channels' existing category information. The main idea of our approach is that, by neighboring the channel overlays, whose category information differs in one dimension, the resulting graph contains enough routing "hints" for seeking other unsubscribed channels' videos. By using SocialVoD, all peers are clustered into a loosely-connected multi-level community structure, which has the small-world property. Based on the new overlay structure, we further propose a routing algorithm for both channel-subscribed users and unsubscribed users. SocialVoD keeps the low maintenance cost feature of previous works, but significantly improves the system's query efficiency, especially when users are developing new watching preferences. Extensive simulation results show the efficiency of our new system in comparison with SocialTube and NetTube.

Our contributions are threefold: (1) We find that the existing P2P systems do not fully utilize the social properties behind the users' watching patterns. Based on the existing channel subscription knowledge and category structure on the Vimeo platform, we introduce a new hierarchical P2P overlay structure, which has the small-world feature. Considering the users' watching patterns also present a small-world phenomenon [10], our new structure is more suitable. (2) Our new system explores the existing category information as an overlay's construction "guide" and routing "hints", which significantly reduces the searching space of videos. We design a new routing algorithm, which creates several node-disjoint searching paths within the P2P overlay. The algorithm can be applied to both channel subscribers and non-subscribers. (3) We provide extensive simulations to show that our new system can efficiently locate the files and is applicable in a large scale.

TABLE I. THE DISTRIBUTION OF VIMEO CHANNELS' ASSOCIATED CATEGORY NUMBERS.

Number of Categories	1	2	3	4	5	...
Percentage (%)	41.94	38.11	8.44	7.54	3.96	0

II. VIMEO CATEGORY AND USER INTERESTS

Papers [1], [8] show that most videos on VoD systems are short, and there are strong clustering behaviors among users. In this section, we further investigate the category features on Vimeo, and the change of users' interests over years.

Vimeo has a total of 23 system-defined categories. By checking the number of channels under each category, we find that there is an obvious gap between a popular group, which has 13 categories, and an unpopular group. We collected the category information of the popular group's channels, each of which has more than 300 subscribers. There are a total of 782 unique channels, and we further compute the distribution of the number of associated categories in Table. I. We find that there are 58.06% channels associated with more than one category feature. Clearly, overlapped categories should be considered during overlay design.

Users may develop new watching preferences over the years. We randomly collect $6,000$ users' channel subscription times, and investigate the change of users' interests (i.e. the category features of subscribed channels), year by year. Table II shows the changing pattern of a typical user's interests. This user subscribed to 87 channels from 2007 to 2011. We divide the records to 5 groups according to subscription time, and count the distribution of the subscribed channels' category features in each year. From the table, we can see that the user becomes more interested in "Narrative" and "Music" related channels, and less interested in the channels from the "Video School" category. Hence, users' watching preferences do change over the years.

Users' watching preferences can be reflected from their channel subscriptions. We summarize the watching pattern of users in the following: (1) Users frequently look for the videos from their subscribed channels; (2) Users are likely to seek other unsubscribed channels' videos, which satisfy their watching habits; (3) The watching habits may change. Users may access certain videos to explore new preferences.

III. OVERLAY DESIGN

A. Feature Coordinate and Feature Distance

More and more VoD systems, such as Vimeo and YouTube, classify video channels by category. Unlike conventional tags, which are created by individual video owners, categories and their subcategories are strictly provided by the system. When a user creates a video channel on Vimeo, he should select several appropriate categories for describing it. Later, the Vimeo staff may add more categories or select suitable subcategory values. Once a channel is created, other users may subscribe to it. We adopt u_i to represent a user, c_i to represent a channel. In this paper, we treat each category as one feature dimension in the

TABLE II. THE CHANGE OF CATEGORY INTERESTS OVER 5 YEARS (%)

Year	Narrative	Arts	Music	Video School	Personal	Big Picture	Sports	Animation	Travel	Tech	Experimental	Comedy	Nature
2011	21.7	16.6	41.7	0	3.3	0	10.0	3.3	0	0	0	3.4	0
2010	41.2	4.8	4.1	4.5	11.4	4.6	0	9.4	13.6	0	1.8	4.6	0
2009	35.7	1.8	12.5	0	0	7.1	14.3	5.4	0	14.3	8.9	0	0
2008	13.0	6.2	22.5	24.5	4.1	4.1	4.1	6.2	0	2.9	4.2	8.2	0
2007	8.8	9.3	8.1	16.8	7.0	0	0	12.5	0	7.6	17.9	6.0	6.0

category-feature space, and we assume that there are a total of m dimensions (i.e. system-defined categories).

The *feature coordinate* of a video channel is defined as $(f_1, f_2, \ldots, f_m)$, where $1 \leq f_i \leq r_i$ and r_i is the number of subcategories in the ith dimension. The physical meaning of value f_i depends on a real system. For example, in the "HD" category dimension, $f_i = 1$ stands for high-definition videos and $f_i = 0$ represents the regular ones, while, in the "Language" category dimension, $f_i = 0, 1, 2, \ldots$ may indicate English, Chinese, Spanish, and et al. Based on the feature coordinates, the category-feature space is partitioned into $\Pi_{i=1}^{m} r_i$ groups, and each channel belongs to one and only one group. We use G_i to represent a hypercube group. Let $\digamma(\cdot)$ be an operation, which gives the feature coordinate of an entity, and $\digamma_i(\cdot)$ gives the coordinate value in the ith dimension: $\digamma(c_i)$ and $\digamma(G_j)$ shows the feature coordinate of channel c_i and hypercube group G_i, respectively.

The *feature difference set* from a source coordinate $S = (s_1, s_2, \ldots, s_m)$ to a destination coordinate $D = (d_1, d_2, \ldots, d_m)$ is represented by $H(S, D) = \{i | s_i \neq d_i, i \in [1, m]\}$. The *feature distance* is measured as the cardinality of the feature difference set $|H|$. For instance, the feature distance set from $(1, 2, 3, 4, 5, 6)$ to $(1, 1, 5, 4, 5, 3)$ is $\{2, 3, 6\}$ since the corresponding values in these three dimensions are different.

B. Channel Subscription and Peer-to-Peer Graph

In practice, a user may subscribe to multiple video channels, the feature coordinates of which may be different from each other. According to the number of subscribed channels, a user u_i is mapped to one or several nodes $\{v_1^i, v_2^i, \ldots, v_j^i\}$ in the P2P network graph. In this section, we assume that each user subscribes to only one channel, and therefore, u_i is uniquely mapped to v_i. The case that users subscribe to more than one channel will be discussed in the next section.

During the construction of the P2P network, a logical connection is created from users u_i to u_j, at the protocol layer, by letting u_i store the IP address of u_j. The whole system forms a P2P network graph $\mathbb{G} = \langle V, E \rangle$, $E \subseteq V^2$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the virtual nodes and $E = \{v_i v_j\}$ gives the logic connection from nodes v_i to v_j. Let $g(c_i)$ be a mapping function from a given channel to a set of nodes, who subscribes to c_i; the inverse function $g^{-1}(v_j)$ gives the channels that v_j subscribed to.

C. Resource Replication Scheme

SocialVoD requires each user to maintain a cache of videos that has been watched within a system-defined session. When the cache is full, the oldest file will be replaced by the new one. This storage scheme can increase the availability of files. Moreover, the popularity of a video changes with time. This caching scheme can eliminate the unpopular files in a timely manner and replace them with the new ones. Here, we are considering a short video, the length of which is usually less than 10 minutes. These files are generally small, and therefore, this cache requirement does not excessively burden users.

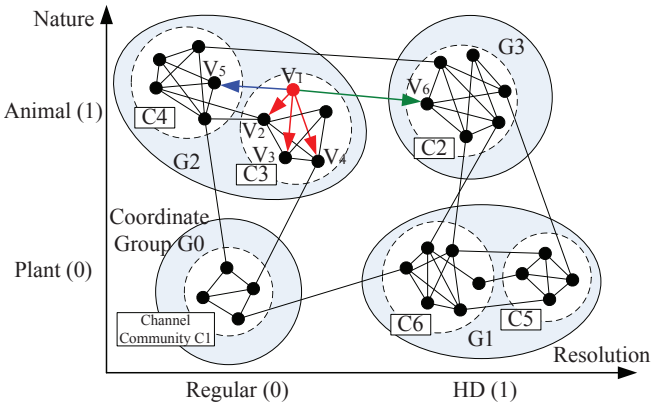


Fig. 2. A diagram of the network structure of SocialVoD.

D. Overlay Construction

When a new node v_i appears, it first sends a request to the server, and the server replies with a set of nodes for establishing logic connections. Suppose v_j is a node from the set. According to the features of v_i and v_j, there are three types of connections in SocialVoD:

- If v_i and v_j subscribe to the same channel $g^{-1}(v_i) = g^{-1}(v_j)$, then edge $v_i v_j$ is an *intra-channel* link.
- If they subscribe to different channels but have the same feature coordinate $g^{-1}(v_i) \neq g^{-1}(v_j)$, $\digamma(v_i) = \digamma(v_j)$, then edge $v_i v_j$ is an *intra-group* link.
- If their feature coordinates differ in one dimension $g^{-1}(v_i) \neq g^{-1}(v_j)$, $H(\digamma(v_i), \digamma(v_j)) = 1$, then edge $v_i v_j$ is an *inter-group* link.

In order to control the usage of resources, SocialVoD limits the number of out-links that a user can create to b. When the server returns the set, it should guarantee the number of connections for each type following intra-channel : intra-group : inter-group $= p : (1-p)q : (1-p)(1-q)$, where p, q are two system-defined parameters and $p, q \in (0.5, 1)$.

Take Fig. 2 as an example. Assume that v_1 is a new peer in the system, who subscribes to channel c_3, $b = 5, p = q = 0.6$ and $m = 2$. When v_1 contacts the server for the first time, the server randomly selects $b \times p = 3$ peers (i.e. v_2, v_3, v_4), who are online and also subscribe to c_3, to build intra-channel links, as shown by red arrows in Fig. 2. The server picks up $b \times (1-p)q \approx 1$ peer from c_4's subscribers, and $b \times (1-p)(1-q) \approx 1$ peer from c_1 and c_3's subscribers to build inter-group links (green arrows) and intra-group links (blue arrows).

The server stochastically creates the connections such that it not necessary to remember the global structure of the overlay graph. When the server seeks the end nodes of the intra-channel links and intra-group links, it randomly picks up the corresponding number of nodes, who are online at the moment. When creating the inter-group links, the server first selects one dimension, in the category-feature space, to be different from it, and then randomly picks up a peer with a different subcategory value in that dimension. This peer is selected as the end of an inter-group link. For a given link, the probability of having two ends with different coordinates is $(1-p)(1-q)/m$. Suppose $F(G_i) = (f_1, \ldots, f_k, \ldots, f_m)$ and $F(G_j) = (f_1, \ldots, f'_k, \ldots, f_m)$. The expected number of out-links from coordinate group G_i to G_j, who differ in the kth category, can be approximated as $b|G_i| \times \frac{|\{v_i|F_k(v_i)=f'_k, i\in[1,n]\}|}{|\{v_j|F_k(v_j)\neq f_k, i\in[1,n]\}|} \times (1-p)(1-q)/m$, where b is the number of out-links per peer, $|G_i|$ gives the number of nodes in hypercube group G_i, and $\frac{|\{v_i|F_k(v_i)=f'_k, i\in[1,n]\}|}{|\{v_j|F_k(v_j)\neq f_k, i\in[1,n]\}|}$ is the probability for selecting f'_k as the new value in dimension k.

SocialVoD creates a two-tier community structure and three levels of overlays. Peers, who subscribed to the same channel, form the low-tier community, also known as the per-channel community C, as illustrated by C_1 to C_6 in Fig. 2. The intra-channel links among the peers build the lowest-level overlay. Because a subscriber is very likely to watch the videos provided by his subscribed channel, clustering peers from the same channel improves files' availability and reduces the average length of query paths. Per-channel communities, who have the same feature coordinate, are clustered into the top-tier community: the coordinate group G, as illustrated by G_0 to G_3 in Fig. 2. The middle-level overlay is established by the links connecting different channels within the same group. SocialVoD connects different groups if their feature coordinates are different in one dimension. The inter-group links form the highest-level overlay. As users tend to access the files with similar features, they may also query the files from unsubscribed channels. SocialVoD puts similar subscribers close to each other, which can improve the query efficiency.

E. Inter-group Link Management

In SocialVoD, a feature-specified query is conducted by forwarding the query messages from group to group via inter-group links. The speed of finding an inter-group link with the specific feature coordinate directly influences the routing efficiency. For accelerating the searching process, the members of each group vote for a leader, who collects the inter-group link information from its members. Usually, the leaders are selected based on their stabilities.

When a node v_i joins a group, he first asks the leader's IP address from his connected peers, who are in the same group. Then, v_i reports all of his inter-group links, together with his own IP, to the leader. When a peer servers as a relay node in an inter-group routing, he should ask the leader for the IP of other members, who have the inter-group links to the next group. The query messages will be forwarded to the next group via these members. Leaders may consume more resources for performing the book-keeping work. We can adopt role rotation and other incentives from the consideration of fairness. When a leader is going to leave SocialVoD, he should find his inheritor and broadcast this update to group members.

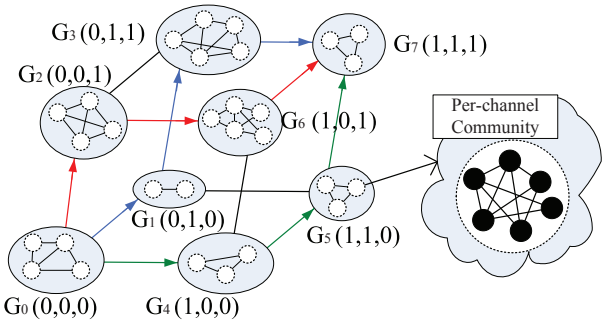


Fig. 3. Coordinate routing sequence-based inter-group routing.

Due to the churn of P2P, peers in SocialVoD periodically check and update their links to other communities. Unlike SocialTube, where peers always update their links to the newest visited peers, SocialVoD removes failed links and replaces them with the links to the other communities, where, recently, the feature-based routing mostly failed. Essentially, SocialVoD gives more emphasis on how to reach a destination, instead of where the destination is.

IV. Routing in SocialVoD

A query is conducted by two steps in SocialVoD. The first step is called feature-spaced inter-group routing, which uses the highest-level overlay to locate the group with the specific feature coordinate. In order to control the network traffic, each query has a limited number of query copies, assuming h. When selecting query paths, one should avoid the situations where more than one query message searches the same space. The second step of a query is to find the target file within the specific group, and we apply the conventional random walk-based approach to conduct the searching.

A. Coordinate Routing Sequence and Node-disjoint Paths

In an m-dimensional hypercube, if a source S has a packet for destination D with feature distance k, there are exactly k node-disjoint shortest paths with length k from S to D. Since these paths uniquely partition the searching space into k non-overlapping subspaces, querying along the node-disjoint paths is both efficient and resource-saving. Suppose that S and D differ in k dimensions, where $H(S,D) = \{1, 2, ..., k\}$.

$R_1 : \langle 1, 2, ..., k\rangle$ is defined as the *coordinate routing sequence*, which determines the resolution order of a given set of dimension differences $H(S,D)$ [11]. For instance, the coordinate sequence R_1 requires peers to construct a query path by gradually solving feature differences from dimensions 1 to k. When $H(S,D) = \{1, 2, ..., k\}$, there are a total of k coordinate routing sequences, and the ith sequence $R_i : \langle i+1, ..., k, 1, ..., i-1, i\rangle$ is created by making $i-1$ circular left shifts of R_1:

- The 1st sequence: $R_1 = \langle 1, 2, ..., k-1, k\rangle$
- The 2nd sequence: $R_2 = \langle 2, 3, ..., k, 1\rangle$
- ……
- The kth sequence: $R_k = \langle k, 1, ..., k-2, k-1\rangle$

Note that the resolution order in R_1 can be any permutation of the elements in feature difference set $H(S,D)$.

Algorithm 1 The selection of coordinate routing sequences at source node S

```
T ← φ
for ∀G_i ∈ N(G_s) do
    for every dimension j ← 1 ... m do
        if F_j(G_i) ≠ F_j(G_s) then
            Compute T'_j, and T ← T ⋃ {(j, T'_j)}
According to T'_j, sort the elements of T in decreasing order.
Select the first h elements and create routing sequences R.
```

Each sequence uniquely defines one shortest path from S to D. Take Fig. 3 as an example. Suppose that G_0 is looking for a video from G_7, and the feature distance set is $H(S,D) = \{1,2,3\}$. When S creates $R_3 = \langle 3,1,2\rangle$, the query path according to R_3 is $G_0 \rightarrow G_2 \rightarrow G_6 \rightarrow G_7$, while, if S creates $R_2 = \langle 2,3,1\rangle$, the query path becomes $G_0 \rightarrow G_1 \rightarrow G_3 \rightarrow G_7$. Hence, the coordinate routing sequence set $R = \{R_1, R_2, ..., R_k\}$ creates k node-disjoint shortest query paths from the feature difference set.

B. The Selection of Coordinate Routing Sequences

In order to control the network traffic, S has only h copies of a query message. In other words, S can conduct h parallel searches for the destination group D. Assuming the feature distance from S to D is k, the relation between h and k influences the selection of query paths. When $h > k$, S is able to use not only the shortest paths for searching D, but also the non-shortest ones. However, if $h < k$, S should determine which path (or coordinate routing sequence) has more of a chance to reach the destination group.

Consider that, for a given category, the popularity of subcategories is different. The sizes of some groups are greater than others. Since the ends of inter-group links are randomly and uniformly selected, a larger-sized group absorbs more inter-group links, and has more of a chance to be visited from other groups. As a result, we approximate a coordinate routing sequence's success rate by subcategories' popularity. Suppose the feature coordinates of S and D are $(s_1, \ldots, s_m)$ and $(d_1, \ldots, d_m)$, and dimension $i \in H(S,D)$. In the highest overlay, the transition probability from any group G, who satisfies $F_i(G) = F_i(S)$, to other groups G', which have the property $F_i(G') = F_i(D)$, is estimated by the following equation:

$$T_i = \frac{|\{v_j | F_i(v_j) = d_i, v_j \in V\}|}{|\{v_j | F_i(v_j) \neq s_i, v_j \in V\}|} \tag{1}$$

where $|\cdot|$ gives the cardinality of a set, and $s_i \neq d_i$. Assume that a path resolves the ith dimension difference first. The successive rate of the path's remaining steps is approximated as follows:

$$T'_i = (|H|-1)! \prod_{j \in H \backslash \{i\}} T_j \tag{2}$$

where H is the abbreviation of the feature difference set $H(S,D)$, and $H\backslash\{i\}$ indicates all the unmatched dimensions between S and D except dimension i.

Algorithm 1 gives the procedure for selecting the coordinate routing sequences when $h < k$. Since the inter-group links are randomly created, a group may not be connected to all other groups, whose feature coordinates differ in one dimension.

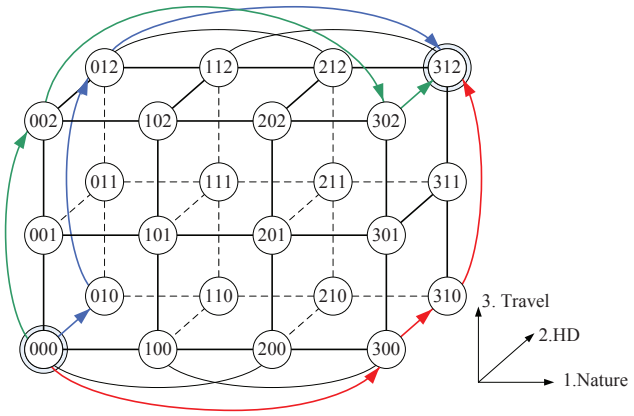


Fig. 4. The hierarchical community structure in SocialVoD.

Assume G_s is the resident group of source S, and let $N(G_s)$ be a set of groups, who are connected with G_s, $N(G_s) = \{G_i | H(F(G_i), F(G_s)) = 1, i \in [1,m], \exists v, v' s.t. v \in G_s, v' \in G_i, vv' \in E\}$. For each neighboring group G_i, Algorithm 1 computes the accessing probability T'_j of the remaining feature difference, if G_i was selected as the first relay group in the routing path (dimension j is the first resolved feature difference). Algorithm 1 creates the corresponding coordinate routing sequences by selecting the h neighbors, who have the largest T'_j. This method essentially explores the rare inter-group links first. The reason is that, as the processing of the feature-based routing, the searching space becomes smaller; fixing the rare dimensions' values first can increase the success rate of the remaining steps.

Take Fig. 4 as an example. Assume that $S = (0,0,0)$, $D = (3,1,2)$, and $H(S,D) = \{1,2,3\}$. The popularity of the 1st dimension "Nature" follows $3:6:10:4$, that of the 2nd dimension "HD" is $4:6$, and that of the 3rd dimension "Travel" is $5:7:3$. There are three potential coordinate routing sequences: $R_1 = \langle 1,2,3\rangle$ (the red path), $R_2 = \langle 2,3,1\rangle$ (the blue path), and $R_3 = \langle 3,1,2\rangle$ (the green path). Using R_1 indicates solving the 1st dimension's difference first. For the source node, the remaining success rate of the red path is estimated as $T'_1 = (3-1)! \times \frac{6}{6} \times \frac{3}{7+3} = \frac{3}{5}$. Similarly, the rates for using R_2 and R_3 can be calculated as $T'_2 = (3-1)! \times \frac{3}{7+3} \times \frac{4}{6+10+4} = \frac{3}{25}$, and $T'_3 = (3-1)! \times \frac{4}{6+10+4} \times \frac{6}{6} = \frac{2}{5}$. If $h = 2$, then the source should use R_1 and R_3.

C. Feature Space Routing

SocialVoD has three types of routings: intra-channel, intra-group, and inter-group. When a user looks for his subscribed channel's videos, an intra-channel routing happens. Query messages are randomly propagated within the per-channel community. Each message is associated with a hop counter, which is decreased by one after each time of forwarding. Once the counter becomes zero, the query message will be discarded. Similarly, if the user seeks for videos from other channels, but have the same feature coordinates, the random walk-based searches are conducted within the corresponding group. However, when the user searches the files from other groups, SocialVoD first sends the messages to the destination group by inter-group routing, and then, adopts intra-group routing to locate the target file within the destination group.

Algorithm 2 Inter-group Routing from Source S

```
1: while h > 0 do
2:   if R ≠ φ then
3:     /*shortest path routing*/
4:     Pick up a sequence R_i from R.
5:     Find G_j ∈ N(G_s) s.t. F_i(G_j) ≠ s_i, F_i(G_j) = d_i.
6:     R ← R\{R_i}, mode ← 0, send (R_i, mode) to G_j.
7:   else
8:     /*non-shortest path routing*/
9:     Find G_j ∈ N(G_s) s.t. F_i(G_j) ≠ s_i, i ∈ H̄.
10:    H̄ ← H̄\{i}, mode ← 1, send (H ⋃ {i}, mode) to G_j
11:  h ← h − 1
```

Algorithm 3 Inter-group Routing for a Relay Node in G_i

```
1: if F(G_i) = F(D) then
2:   Use intra-group routing within G_i.
3: else
4:   if mode = 0 then
5:     R ← R\R[1], the next resolving dimension k ← R[1].
6:     Find G'_i ∈ N(G_i) s.t. F_k(G_i) ≠ d_k, F_k(G'_i) = d_k.
7:     Send (R, mode) to G'_i.
8:   if mode = 1 then
9:     H ← H\{j}.
10:    Find G'_i ∈ N(G_i) s.t. ∃k ∈ H, F_k(G_i) ≠ d_k, F_k(G'_i) =
       d_k. Send (H, mode) to G'_i.
```

The inter-group routing at a source node is given by Algorithm 2. Based on the relation between h and k, there are two modes, where $mode$ is 0 for a shortest path and 1 for a non-shortest path. Assume that the feature coordinate of source S is $(s_1, \ldots, s_m)$ and that of D is $(d_1, \ldots, d_m)$. The feature difference set is represented by $H = \{i | s_i \neq d_i, i \in [1, m]\}$. The coordinate sequences set $R = \{R_1, R_2, ..., R_k\}$ provide the basic guidance for the shortest path mode. Let $\overline{H}$ be the compliment set of H, $\overline{H} = \{i | s_i = d_i, i \in [1, m]\}$, which will be used for the non-shortest path mode.

Algorithm 2 gives the shortest path routing with a high priority; as long as set R contains some coordinate sequences (i.e. $h \leq k$), S should use the corresponding shortest path first. S picks up a sequence R_i from R. Suppose the first resolving dimension of R_i is i. S finds G_j, whose ith dimension value is the same as that of D, from neighboring groups $N(G_s)$. Then, S sends R_i to G_j and asks it to use shortest path mode. When R becomes empty (i.e. $h > k$), it switches to non-shortest path mode, in which the source node increases the feature distance from k to $k+1$. For instance, in Fig. 3, G_0 is sending message to G_3, where $F(G_0) = (0, 0, 0)$ and $F(G_3) = (0, 1, 1)$. Although there are only two different dimensions, S can first send the message to group $(1, 0, 0)$ and then forward it to the destination. The path becomes $G_0 \rightarrow G_4 \rightarrow G_6 \rightarrow G_7 \rightarrow G_3$.

Algorithm 3 is designed for relay nodes. Assume the current relay node is in group G_i, and the node got the query message from G_0, where $G_i \in N(G_0), F_j(G_i) \neq F_j(G_0)$. Let $R[1]$ be the first element in the coordinate routing sequence R. If G_i is the destination group, then intra-group routing will be used to search the file. When G_i is not the destination and the routing is in the shortest path mode, the dimension is different, and has been resolved from G_0 to G_i; it should be removed (step.5). The query message is forwarded to the next group according to the new head of R. As for the non-shortest path mode, any element from the dimension difference set can be used as the next resolving dimension. For example, in Fig. 3, G_0 sends $H = \{1, 2, 3\}$ and $mode = 1$ to G_4. At G_4, the unsolved dimension set becomes $H = \{2, 3\}$, and G_4 can forward the message either to G_5 or G_6. One can use Equation. 2 to find out an optimal receiving group.

V. Extension

A. Multi-channel Subscriptions

The main idea of SocialVoD is to cluster users according to their subscribed channels' category features. However, since a user may subscribe to several channels with different feature coordinates, if we directly use the basic scheme of SocialVoD, peers will form into one giant community, instead of several groups. For instance, users u_1 and u_2 are the members of channel C, and users u_2 and u_3 subscribe to another channel C'. Directly connecting u_1 u_2 and u_2 and u_3 will eventually put u_1 u_3 into the same community. To avoid this situation, a user is mapped to several nodes on the P2P graph, according to his own subscription. Suppose user u_i subscribed to j channels, then u_i is mapped to a set of nodes $\{v_1^i, v_2^i, \ldots, v_j^i\}$. Each node is associated with b/j out-links (the number of out-links for each user is still b). When searching a file, the user should decide to use which node v_k^i as the source node, since they may have different feature distances to the destination group, as illustrated by Fig. 5.

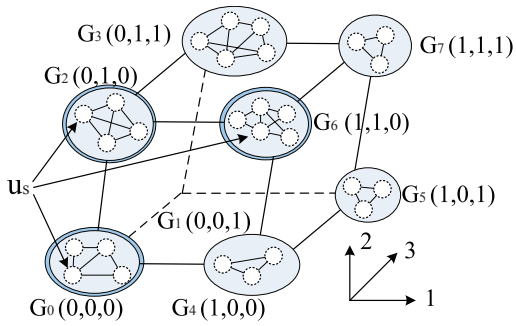


Fig. 5. Source selection when a user subscribes to multiple channels.

Algorithm 4 gives the procedure for selecting the initial node of a routing. The idea of this scheme comes from Algorithm 1. Let $M(u)$ be the mapping function from user u to his corresponding nodes on the graph, $M(u) = \{v_1, v_2, \ldots, v_j\}$. Basically, Algorithm 4 tests all elements in $M(u)$, and calculates the remaining path's success rate T' from these nodes. Then, it selects the top h paths with the highest success rate. Note that, in practice, not all groups, whose feature coordinates differ in one dimension, are connected. Also, some groups may temporarily not contain any peer (all of them are off-line.) As a result, the nodes, which are closer to the destination in the feature space, may not always be the best source.

Let us go back to Fig. 4's example, where $D = (3, 1, 2)$, and the popularity of each dimension follows $3 : 6 : 10 : 4$, $4 : 6$, and $5 : 7 : 3$. Suppose that the user also subscribes to a channel in group $(2, 1, 1)$, then the feature difference set is $H = \{2, 3\}$. There are two potential coordinate routing sequences: $R_1 = \langle 1, 2 \rangle$ and $R_2 = \langle 2, 1 \rangle$. If we solve the 1st dimension first, then we have $T'_1 = \frac{1}{5}$, while, if we solve the

Algorithm 4 The selection of initial node for a user u

```
T ← φ
for Every v ∈ M(u) do
    Find G_s s.t. F(G_s) = F(v).
    for ∀G_i ∈ N(G_s) do
        for j ← 1...m do
            if F_j(G_i) ≠ F_j(G_s) then
                Compute T'_j, and T ← T ⋃ {(v, j, T'_j)}
According to T'_j, sort the elements of T in decreasing order.
Select the first h elements in T, and use them as routing sources.
```

2nd dimension, we get $T'_2 = \frac{7}{10}$. Again, if $h = 2$, then the user should use routing sequence $\langle 1, 2, 3\rangle$ from group $(0, 0, 0)$ and $\langle 2, 1\rangle$ from group $(2, 1, 1)$.

When users are allowed to subscribe to multiple channels, the inter-group routing can be accelerated. Here, we propose a new approach, called *feature matching shortcut*. In the classic hypercube routing, each instance of message forwarding can solve only one dimensional difference. But, in SocialVoD, since a user is mapped to several nodes, it is possible that the physic holder of a query message may be mapped to another peer, who is more than one feature distance closer to the destination. Take Fig. 5 as an example. Suppose v_i from G_2 gets a message with G_6 as destination. Since there is another node v_j possessed by the same user u, $v_i, v_j \in M(u), v_j \in G_6, v_i \in G_2$, u can stop the inter-group routing and begin to use intra-group routing from v_j's out-links.

B. Routing for non-subscribers

As for users, who have not subscribed to any channel, there is no feature coordinate information. Based on historical query records, SocialVoD deducts the most frequently searched subcategory features of a non-subscriber, in each dimension, and uses these values as the user's feature coordinate. Inter-group links are generated from the user to the corresponding coordinate group, which consists of the subscribers, who have the same feature coordinate. Since non-subscribers' coordinates are not accurate, links are created only from a non-subscriber to subscribers, and not vice versa. When a non-subscriber wants to watch some videos, the inter-group routing will be conducted from the linked coordinate group. Once a non-subscriber's watching preference is formed (i.e. subscribe to certain channels), most of his links will have already been connected to the subscribed communities.

C. Pre-fetch for new videos

One common problem for new videos is their availabilities. Unlike other VoD platforms, Vimeo allows all subscribers of a channel to upload their videos. After a new video is being uploaded, in the initial phase, most queries about it are responded to by the central servers. For improving the availabilities of new videos, SocialVoD uses a pre-fetch scheme, which pushes the new contents to other peers from the same per-channel community. Whenever a new video is generated, the source node sends out several random walkers, carrying the video, to his resided community. When a walker stops, the latest visited peer will store the video in cache.

VI. Performance Analysis and Evaluation

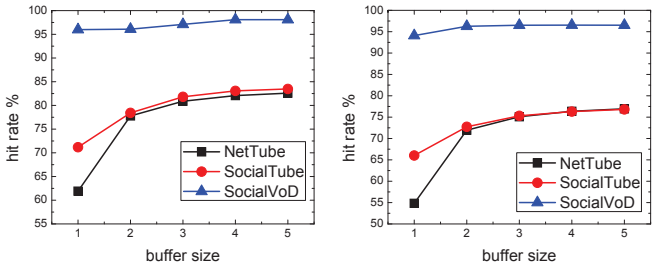
In this section, we conduct extensive simulations to evaluate the performances of our proposed system by using both synthetic data and real data. For the ease of comparison, we call our scheme, which clusters the subscribers of the same channel into a community and connects different communities, whose coordinates differ in one dimension, *SocialVoD*; the previous work [1], which only clusters the peers in individual categories, is called *SocialTube*; we also compare the results with *NetTube* [8]. Since these approaches use different searching schemes, we keep the maximum possible length of their overall searching paths be the same.

A. Simulation Setup and Evaluation Metric

To measure the performance of SocialVoD, we conducted extensive simulations by using Matlab. For the synthetic simulation, we assume that there are 4 categories and each category has 4 different value types (i.e. subcategories); For the real one, we use the real category features of Vimeo channels. We generate $5,000$ peers for the synthetic simulations and $15,000$ peers for the real ones. Each peer is associated with a fixed-size buffer and outlink budget. While setting up, we randomly assign a popularity degree to each subcategory, and each peer randomly selects the subscribed channel coordinates based on the popularity. Therefore, different channels have different sizes. During simulation, some peers randomly create new videos and save into their own buffers.

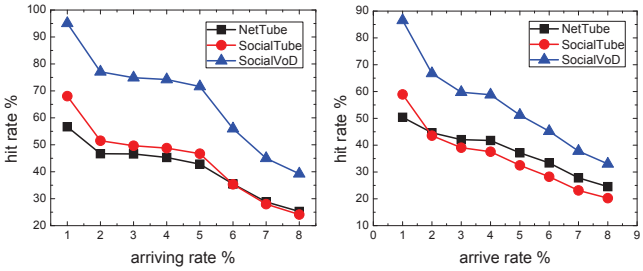
In this paper, we emphasize on the following two aspects of users' watching behaviors: 1) frequently watching their subscribed channels' videos; 2) developing new watching preferences by exploring other unsubscribed channels that have similar features. To accurately simulate the watching behaviors, we use the following video selection mechanism. For each dimension, we assign users with a probability for watching the videos with the same subcategory feature. As a result, a user may query the videos from his subscribed channel, and he may also query for other types of videos; users have a better chance of the videos with similar features to their subscribed channels than a totally different video type. After sending a query, a node will wait for a TTL. When time is up, the node will send a query directly to the central server and get the video file. No matter whether the node found the corresponding video from the P2P system or not, eventually, it will get the file and locally save it in the buffer.

When building the P2P overlay in the feature space, the number of outlinks from one community to the other is determined by the number of peers subscribed to the channel and system variable $(1 - p)(1 - q)$. During feature space routing, SocialVoD sends out several query messages with different coordinate sequences. Since the feature space overlay is stochastically established, some messages may not be able to reach the communities with the target feature coordinates. Once a message reaches the destination communities, it uses a random walk-based scheme to search the videos in the buffers of the community's peers. Later, we test the impacts of message copy numbers on both feature space routing and inner community routing. In our simulation, we are interested in the following two metrics: (1) Average hitting rate: the percentage of queries having found the corresponding videos from P2P



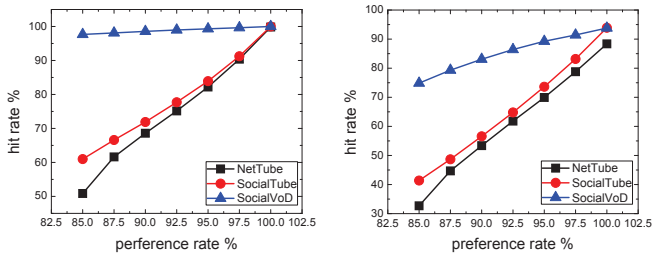
(a) Uniform distribution (b) Normal distribution

Fig. 6. The impact of the buffer size



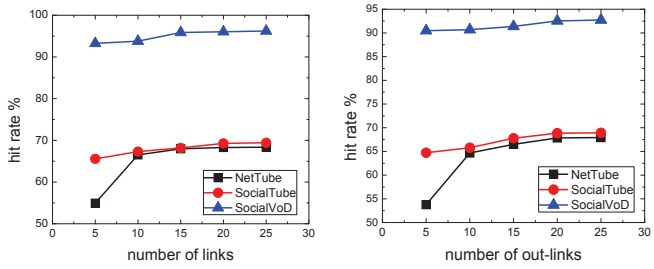
(a) Uniform distribution (b) Normal distribution

Fig. 7. The impact of the new content's arriving rate



(a) Uniform distribution (b) Normal distribution

Fig. 8. The impact of users' watching preferences



(a) Uniform distribution (b) Normal distribution

Fig. 9. The impact of each peer's total number of links

overlays. This value shows the percentage of workloads that have been reduced from the server. During simulation, we mainly consider this metric. (2) Average delay: the average waiting time before watching a video. Note that all three approaches first search the videos on P2P overlays, and then resort to the server. Hence, the delay has a upper bound value. Generally speaking, the shorter the average delay is, the better a P2P overlay design is.

B. Simulation results

We first investigate the impacts of the buffer's size. The buffer temporarily stores the videos that have been watched before. Clearly, the bigger the buffer is, the higher the probability is for finding a video by P2P overlay. However, the benefits for using buffers do not grow linearly. In this simulation, we let a peer's query probability be 80%, different dimension's exploring probability be 35%, let 5% of peers generate new contents per simulation session (with there being 20 sessions). As shown by Fig. 6, with the growth of the buffer, the average hitting rates go up. SocialVoD always has the highest average hitting rate. Moreover, when the popularity of subcategory values follows normal distribution, the hitting rate of SocialVoD is lower than that of the uniform distribution condition. However, SocialVoD still beats others.

In Fig. 7, we consider the impacts of the new contents' arriving rates. If a lot of new videos are generated in a short time, only a few peers may cache the files. As a result, with the growth of the new content arrival rate, the average hitting rates quickly drop down, which is much faster than the changing speed of the average delay. In simulation, we let the generation rate of each peer change from 1% to 8%. We assume that peers know about the new contents. As more contents are generated, users have more choices for the queried contents. If a node's query frequency is slower than the system's new content's generating frequency, the cached files will soon become useless. Here, we let the average query rate of each peer be 75% per session, and the size of a peer's buffer be 1.

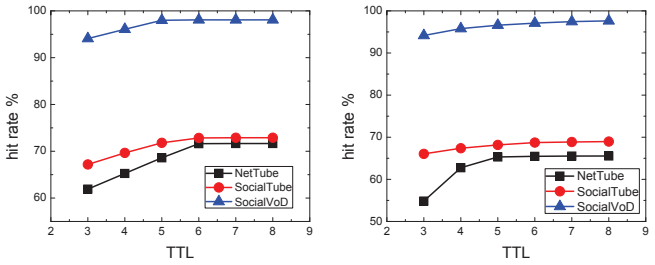
Next, we study the influence of the users' watching preferences. Based on the feature coordinates of users' subscribed channels, each user has his own watching preferences. However, any user may also explore other types of videos with similar feature coordinates. In Fig. 8, we gradually decrease the probability of exploring new features for each query from 18.5% per dimension to 0%, and we can see that the average hitting rate of all methods goes up. Note that each video has several dimensions. For example, if the exploring probability is 30% and there are 4 dimensions, the query rate for a peer's own subscribed channel's video is about 24%. Since SocialVoD is specially designed for exploring other video features, its performance is much better than that of others, especially when the system's dimension exploring rate is high.

The number of links that each peer connects to essentially affects the query's efficiency. A well-connected community has a relatively short mixing time, and the random walk-based queries from the same peer can quickly become independent of each other. Moreover, for the overlay in feature space, having more outlinks from a peer can increase the routing success rate. In Fig. 9, we gradually increase the number of outlinks per node from 5 to 25, and keep 10% of the outlinks connecting to other communities. We can see that the average hitting rate gradually becomes higher and higher. Note that, due to new content existing, and the stochastic feature of our overlay structure, the average hitting rate cannot reach 100%.

In the next two figures, we study the impacts of the length of the searching paths during the random walk-based query. For SocialVoD, when there are less query budgets for inter-community routing, it will only use the shortest paths' coordinate sequences. However, if there are more query paths, then the SocialVoD algorithm can also adopt the coordinates of non-shortest paths. In general, the number of non-shortest paths is much greater than that of the shortest paths. So, the success

TABLE III. THE IMPACT OF THE PUSH OPERATION FOR NEWLY GENERATED CONTENTS.

Method	SocialVoD	SocialVoD (push)	SocialTube	SocialTube (push)
Delay	14.369	14.003	26.121	25.912
Hit rate	97.39%	98.35%	77.10%	77.63%



(a) Uniform distribution (b) Normal distribution

Fig. 10. The impact of the length of random walks.

rate of inter-community will be increased by using more query messages. For all approaches, including SocialVoD, SocialTube and NetTube, the hit rate is also related with the length of intra-community random walks. In the simulation, we gradually increase the query's TTL of intra-group routing from 3 to 8, and keep these approaches' maximum possible overall length of query paths be the same. Fig. 10 gives the results. Because SocialVoD first uses a relatively small number of messages to shrink the searching space of potential peers, it performs better than others in the aspects of hitting rate. Note that, since SocialTube is specially designed for YouTube where each channel can only belong to one and only one category, its intra-category searching becomes less efficient when channels have multi-dimensional features.

In order to increase the average hitting rate, whenever a peer generates a new content, he can push the data to other peers randomly. Essentially, the operation can increase the availability of the new contents, and there are several ways to implement it. For example, from the new content owner, he could send out several copies of the new content and use random walk-based approach to spread the content within his community. Here, we test the simplest case, where the new content is directly pushed to the peer's neighbors. Table. III shows that the pushing scheme can improve the new contents' availability on a P2P overlay. Both SocialVoD and SocialTube have less average delays for the queries of the new contents.

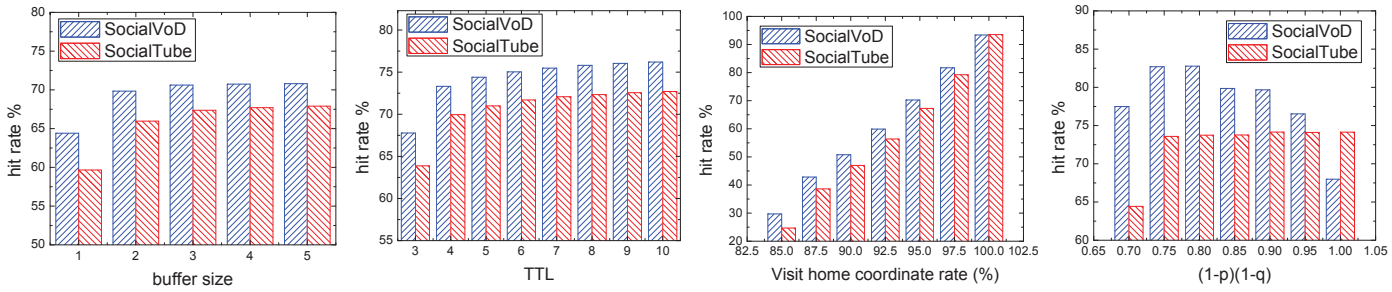
Fig. 11 shows the testing results by using real data. We collect the category information of Vimeo channels, who have more than 300 subscribers. In total, there are 782 channels and 13 categories. Users are randomly assigned to different channels and the average number of users is 12.7. The experiment results of Figs. 11 (a), (b), and (c) are consistent with that of the synthetic simulations. In Fig. 11 (d), we test the impacts of the value of $(1-p)(1-q)$, which controls the percentage of outlinks connecting to other communities that have different feature coordinates. There is a trade-off between the number of inter-community links and intra-community links. With the growth of intra-group (or inter-group) links, searching files from the resident community (or other communities) becomes easier. From Fig. 11 (d), we find that both approaches are affected by p and q; However, SocialVoD gets more influence. The reason for this phenomena is that, when looking for videos with different features, SocialVoD utilizes the inter-group links for routing, while, SocialTube independently seeks files from each related dimensions. As a result, SocialVoD gets more impacts when there is few inter-group links (i.e. $(1-p)(1-q)=1$). But, for most cases, SocialVoD still has better hit rate than SocialTube.

VII. RELATED WORK

The research on P2P architecture has existed for a decade. Many papers have proposed different schemes [12]–[15]. PA-VoD [7] and NetTube [8] are the two most classic unstructured-P2P systems. PA-VoD builds a video's overlay by grouping all the peers, who are watching the same video, together. When a user finishes watching, he no longer acts as a provider and eventually will be wiped out from the overlay. PA-VoD works, if there are plenty of users watching the same long video, such as a movie, the typical length of which is 1-2 hours. However, as for the VoD platform, such as Vimeo, most videos are short, the median length of which is about 200 seconds [16]. As a result, a majority of short videos' requests are handled by the central server, instead of peers. The basic idea of the NetTube is that the users, who are watching the same videos, are more likely to view the same content in the future. By associating a cache buffer with each node, NetTube puts the viewers of the same video into one overlay, and further allows them to search for videos from each other. Unlike PA-VoD, the users of NetTube will temporarily store the videos, which were watched previously, in their local buffer. Therefore, after he or she finishes watching a video, as long as the buffer contains the file, the user will remain in the video's overlay. This unique design significantly improves the resource availability in a P2P system. However, both NetTube and PA-VoD create an overlay for each video, which wastes plenty of network resources.

Hypercup [9] is a typical structured-P2P system, which organizes peers into a hypercube structure. Compared to the unstructured system, Hypercup is fault tolerant and suitable for efficient search. However, since Hypercup constructs the overlay by directly connecting peers to peers, as for high churn conditions, where peers frequently join and leave the system, it suffers from huge maintenance traffic. Unlike Hypercup, SocialVoD treats a group of users, who have the same watching preferences, as a node in hypercube. Although a system may face the frequent connection and disconnection of peers, the absence of a whole group is rare. As a result, SocialVoD has a high searching efficiency and low maintenance costs. The idea of node group-based hypercube has existed for a while [17], [18]. However, the existing schemes neither consider the social property of peers [19], nor are they suitable for short videos; they simply connect all peers of a group to a super-peer, and use the super-peers to build a hypercube structure. Since the size of a VoD channel varies greatly from tens of subscribers to millions, and there are plenty of channels having the same category features, the conventional scheme does not suit the modern VoD system any longer. SocialVoD clusters the peers, who subscribe to the same channel, into a community, and further puts similar communities close to each other.

The design of P2P overlay's structure is highly related with users' data accessing pattern. Unlike traditionally streaming video systems, modern VoD systems, such as Vimeo and YouTube, have social aspects [20]. As shown by papers [10], [21], the date-sharing patterns in VoD systems have a small



(a) buffer size (b) length of random walk (c) watching preference (d) community structure

Fig. 11. Real data-based simulation results.

world property, which suggests that the videos on these systems have strong correlations with each other. As a result, the P2P overlay structures of VoD should also possess the small world property. A theoretical model for small-world networks by paper [22] picture a small world as a loosely connected set of highly-connected subgraphs [10]. The structure of our proposed SocialVoD overlay exactly satisfies this structure.

VIII. Conclusion

Over the past decade, we have witnessed an explosive growth of online video-on-demand (VoD) services, such as Vimeo and YouTube. With the growth of the services' popularity, the scalability, bandwidth, and delay problems of the conventional client-server architecture have become increasingly obvious, which has compelled the development of the peer-to-peer architecture. In this work, we leveraged the existing social relationships (i.e. common channel subscription) and social similarity (i.e. the similarity of users' watching preferences) to establish a new P2P overlay structure. Unlike other VoD systems, the users on Vimeo have more social interactions: any subscriber of a channel is allowed to upload his videos to the channel, and all subscribers can watch and discuss the videos' contents within the channel. Clearly, in such a high-interaction platform, a channel's subscribers are likely to watch the videos from the same channel. Moreover, users' watching preferences gradually vary, and therefore, they also tend to watch other unsubscribed channels' videos, which have similar content features (i.e. category values). Based on these properties, our system, SocialVoD, creates a hierarchical overlay among peers: subscribers of the same channel form into a low-level community, and in high-level overlay, different channels' groups are connected based on their similarity. This unique structure has three significant features. First, the SocialVoD provides relatively short query paths when users are searching for the videos, the features of which are close to their subscribed channels. The second advantage is that SocialVoD can efficiently locate the low-level communities with specific features via the high-level overlay structure. Third, compared to the conventional per-channel overlay structure, in SocialVoD, each peer has low maintenance costs. Extensive simulation results show the superior performance of our approach.

Acknowledgment

This research is supported in part by NSF grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963.

References

[1] 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 *IEEE ICDCS*, 2014.

[2] "YouTube Statistics," https://www.youtube.com/yt/press/statistics.html, accessed: 2014-10-20.

[3] "Vimeo Wikipedia," http://en.wikipedia.org/wiki/Vimeo, accessed: 2014-11-16.

[4] "5 Reasons to Choose Vimeo," https://www.youtube.com/yt/press/statistics.html, accessed: 2014-11-16.

[5] "Top 10 Most Popular YouTube Channels by Subscribers," http://www.ignitesocialmedia.com/youtube-marketing/top-10-most-popular-youtube-channels/, accessed: 2014-09-16.

[6] J. Lu and J. Callan, "Content-based peer-to-peer network overlay for full-text federated search," in *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*. C.I.D., 2007.

[7] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?" *ACM SIGCOMM*, 2007.

[8] X. Cheng and J. Liu, "Nettube: Exploring social networks for peer-to-peer short video sharing," in *IEEE INFOCOM*, 2009.

[9] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuPhypercubes, ontologies, and efficient search on peer-to-peer networks," in *Springer AP2PC*, 2003.

[10] A. Iamnitchi, M. Ripeanu, E. Santos-Neto, and I. Foster, "The small world of file sharing," *IEEE TPDS*, 2011.

[11] H. Zheng, Y. Wang, and J. Wu, "Optimizing Multi-copy Two-hop Routing in Mobile Social Networks," in *IEEE SECON 2014*, 2014.

[12] S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Massoulie, "Providing video-on-demand using peer-to-peer networks," in *IPTV workshop*, 2006.

[13] B. Cheng, L. Stein, H. Jin, X. Liao, and Z. Zhang, "GridCast: Improving peer sharing for P2P VoD," *ACM TOMCCAP*, 2008.

[14] B. Li, M. Ma, Z. Jin, and D. Zhao, "Investigation of a large-scale P2P VoD overlay network by measurements," *Springer PPNA*, 2012.

[15] E. Rosas, N. Hidalgo, M. Marin, and V. Gil-Costa, "Web search results caching service for structured P2P networks," *FGCS*, 2014.

[16] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *ACM SIGCOMM IMC*, 2007.

[17] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser, "Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks," in *ACM WWW*, 2003.

[18] A. Löser, W. Siberski, M. Wolpers, and W. Nejdl, "Information integration in schema-based peer-to-peer networks," in *Springer AISE*, 2003.

[19] K. Lewis, M. Gonzalez, and J. Kaufman, "Social selection and peer influence in an online social network," *PNAS*, 2012.

[20] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *IEEE IWQoS*, 2008.

[21] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: A YouTube-based measurement study," *IEEE TMM*, 2013.

[22] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, 1998.