SocialTube: P2P-assisted Video Sharing in Online Social Networks

Haiying Shen*, *Senior Member, IEEE*, Ze Li, *Student Member, IEEE,* Yuhua Lin and Jin Li, *Fellow, IEEE*

Abstract—Video sharing has been an increasingly popular application in online social networks (OSNs). However, its sustainable development is severely hindered by the intrinsic limit of the client/server architecture deployed in current OSN video systems, which is not only costly in terms of server bandwidth and storage but also not scalable with the soaring amount of users and video content. The peer-assisted Video-on-Demand (VoD) technique, in which participating peers assist the server in delivering video content has been proposed recently. Unfortunately, videos can only be disseminated through friends in OSNs. Therefore, current VoD works that explore clustering nodes with similar interests or close location for high performance are suboptimal, if not entirely inapplicable, in OSNs. Based on our long-term real-world measurement of over 1,000,000 users and 2,500 videos on Facebook, we propose SocialTube, a novel peer-assisted video sharing system that explores social relationship, interest similarity, and physical location between peers in OSNs. Specifically, SocialTube incorporates four algorithms: a social network (SN)-based P2P overlay construction algorithm, a SN-based chunk prefetching algorithm, chunk delivery and scheduling algorithm, and a buffer management algorithm. Experimental results from a prototype on PlanetLab and an event-driven simulator show that SocialTube can improve the quality of user experience and system scalability over current P2P VoD techniques.

Index Terms—Video-on-demand (VoD), On-line social networks, Peer-to-peer networks, Peer-to-peer assisted VoD.

1 Introduction

Online social networks (OSNs) (e.g., Facebook, Twitter) are now among the most popular sites on the Web. An OSN provides a powerful means of establishing social connections and sharing, organizing, and finding content. For example, Facebook presently has over 500 million users. Unlike current file or video sharing systems (e.g., BitTorrent and YouTube), which are mainly organized around content, OSNs are organized around users. OSN users establish friendship relations with real-world friends or virtual friends, and post their profiles and content such as photos, videos, and notes to their personal pages.

Video sharing has been an increasingly popular application in OSNs, enabling users to share their personal videos or interesting videos they found with their friends. Indeed, according to comScore Releases in August 2010, Facebook is now the second-largest online video viewing platform. The total time spent on video viewing on Facebook increased 1,840% year-over-year, from 34.9 million minutes in October 2008 to 677.0 million minutes in October 2009. During the same time period, the number of unique video viewer increased by 548% and total number of streams grew by 987% [1].

The recent rapid development of OSN video sharing applications illustrates the evolution of OSNs from simply communication focused tools to a media portal. OSNs are transforming from a platform for catching up with friends to a venue for personal expression and for sharing a full variety of content and information. However, OSN's further advancement is severely hindered by the intrinsic limits of the conventional client/server architecture of its video sharing system, which is not only costly in terms of server storage and bandwidth but also not scalable with the soaring amount of users and video content in OSNs. For example, the world's largest video sharing website, YouTube, spends roughly $1,000,000/day for its server bandwidth [2]. This high and ever rising expense was one of the major reasons that YouTube was sold to Google. OSNs are now facing the same formidable challenge as YouTube, as more and more users rely on Facebook for video sharing. Though OSNs can depend on content delivery networks (CDNs) for video content delivery (e.g., Facebook depends on Akamai for video delivery), the CDN service is costly.

In recent years, much effort has been devoted to improving the client/server architecture for video sharing, with the peer-to-peer (P2P) architecture being the most promising. P2P-based video sharing has been used in on-demand video streaming (e.g., GridCast and Vanderbilt VoD). With each peer contributing its bandwidth to serving others, the P2P architecture provides high scalability for large user bases. Previous P2P VoD systems either randomly cluster peers for video inquiry [3]–[18] or form certain peers into a distributed hash table (DHT) for chunk indexing [19]–[21]. In order to reduce the video transmission and/or prefetching delay, some works cluster nodes with close physical proximity [7], [8], [19] or similar interests [9], [21]. However, those mechanisms are suboptimal, if not entirely inapplicable, in OSNs. Unlike VoD systems that provide system-wide video searching and sharing, where a peer can access any other peer's content, OSNs do not provide video search functionality. In an OSN, videos are visited and spread by the users' friends through the Friend-of-Friend (FOF) relationship. Specifically, a friend of user X, say user Y, receives notification when user X uploads a video, and user Y's friend Z is informed about the video only

- * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*
- *Haiying Shen, Ze Li and Yuhua Lin are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634. E-mail: {shenh, zel, yuhual}@clemson.edu*
- *Jin Li is with Microsoft research, Redmond, WA 9805. E-mail: jinl@microsoft.com*

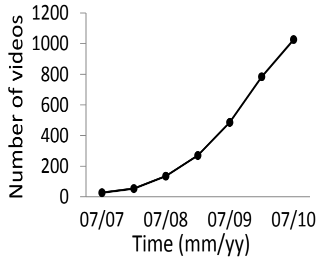


Fig. 1: Number of videos uploaded.

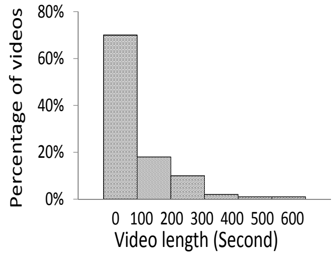


Fig. 2: Distribution of video length.

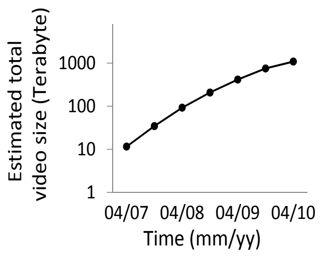


Fig. 3: Estimated total size.

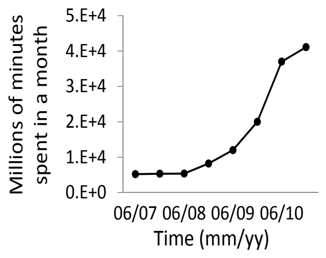


Fig. 4: Time spent on Facebook.

when user Y "shares" the video, and so on. Facebook has options to share videos only between friends, FOFs (default), or everyone. Therefore, users in an OSN watch videos driven much more by the friendship relation than video content.

In order to investigate the video watching behaviors of users in OSNs, we queried more than 1,000,000 users and retrieved about 2,500 public visible videos metadata on Facebook. Our measurement reveals that (1) most of the viewers of a user's videos are the user's close friends, (2) most video views are driven by social relationships, and the rest are driven by interests, and (3) viewers of the same video tend to reside in the same location. Based on our observations, we propose SocialTube, a system that explores the social relationship, interest similarity and location to enhance the performance of video sharing in OSNs. Specifically, an OSN has a social network (SN)-based P2P overlay construction algorithm that clusters peers based on their social relationships and interests. Within each cluster, nodes are connected by virtue of their physical location in order to reduce video transmission latency. SocialTube also incorporates an SN-based chunk prefetching algorithm to minimize video playback startup delay. We have conducted extensive experiments in an event-driven simulator and implemented a prototype on PlanetLab to evaluate the performance of SocialTube. Performance results show that SocialTube greatly reduces the workload of the server, improves the quality of playback, and scales well to a large client population. In the supplemental material, we present a chunk delivery and scheduling algorithm and a buffer management algorithm to further enhance the performance of SocialTube.

To our knowledge, this work is the first that studies the distinct characteristics of OSN video sharing that vary from other content-based system-wide video sharing, and builds a P2P-based video sharing system in an OSN by leveraging those characteristics for higher performance. Our previous conference version of this article [22] introduces the basic trace data analysis and design of SocialTube. This article presents more trace data analytical results. It also presents new SocialTube mechanisms including locality-aware video prefetching mechanism, two policies to increase the chunk delivery abilities, and buffer management algorithm. This article further presents more simulation results and the experimental results for the SocialTube prototype on the real-world PlanetLab testbed.

The rest of the paper is organized as follows. Section 2 and Section 3 present the Facebook trace data analysis and the design of SocialTube. Section 4 and Section 5 present the trace-driven simulation results and experimental results on PlanetLab. We conclude this paper with remarks on future work in Section 6. The supplemental material presents two enhanced methods including chunk delivery and scheduling and buffer management algorithm. It also presents additional experimental results and an overview of related work.

2 Facebook Measurement and Analysis

In this section, we present our Facebook trace measurement results and give an in-depth perspective of Facebook video viewing patterns, that shows the necessity of peer assistance in OSN video sharing and provides a direction for the design of a P2P video sharing system in OSNs.

We used breadth-first-search [23] to query over 1,000,000 users seeded by 5 users in the USA. In order to avoid overloading the Facebook, we sent a query to Facebook every 5s. We can only see the video activities of the users who are friends or FOFs of the crawler and the users that chose "everyone" as their video access option. Because of this access limit, we only found about 2,500 videos and 12,000 users who watched these videos during the time period from Jul. 2007 to Aug. 2010, which is used as a sample for the video sharing and watching activities. The collected dataset includes the information about user friendship relations, interests, locations, and videos uploaded and shared by users. For each video, we retrieved the video metadata such as its title, length, and viewers when available. To respect the privacy of the users, we anonymize the user names before storing the data in our database. We only crawled the video metadata of these users, with other personal information untouched.

2.1 Popularity of Videos on Facebook

First, we investigate the popularity of videos on Facebook over the years. Figure 1 plots the number of videos corresponding to the time they are uploaded in our collected video pool. It shows that the number of videos uploaded to Facebook increases sharply along with time. Since Facebook launched its video service in 2007, the increasing trend of video uploading has never slowed down, making it one of most popular applications on Facebook. Figure 2 shows the video length distribution in our collected video pool. From the figure, we can see that about 70% of the videos are less than 100 seconds. Videos longer than 200 seconds account for less than 10% of all videos. It may be because users generally share short user-generated videos of their lives with their friends in OSNs. Based on the measured results, we make an observation (O):

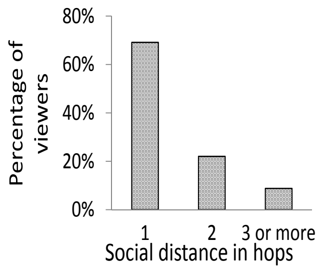


Fig. 5: Social distance.

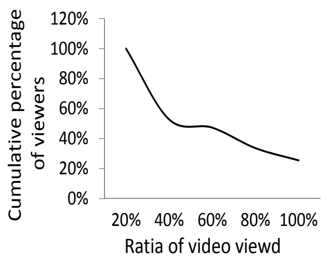


Fig. 6: Different viewer types.

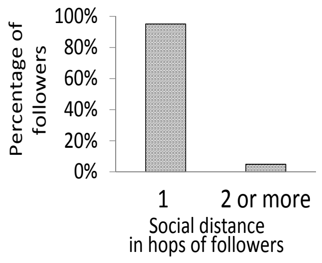


Fig. 7: Social distance between follower and video owners.

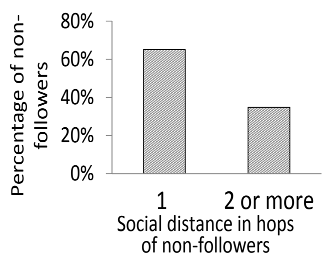


Fig. 8: Social distance between non-follower and video owners.

O1: *Video sharing in Facebook is becoming increasingly popular, and most are very short.*

Typical Facebook videos have a bitrate of [300, 600] kbps. Based on the statistics from [1], the total number of videos on Facebook was 41,537,000 in April, 2009. Using the increasing trend derived from Figure 1, we calculated the approximate number of videos on Facebook at different times. Using the average video length calculated from Figure 2, we obtain an estimate of the server storage cost for videos as shown in Figure 3. The figure indicates that the storage cost of videos has increased 100 times from 10 Terabytes to 1,000 Terabytes, which is a great burden for the server considering the high bandwidth cost caused by uploading and downloading videos.

O2: *The bandwidth and storage burden on video servers for providing video services is high and has been increasing at a rapid rate.*

According to the comScore [1] study, users on average spent 9.9% of their total online time on Facebook. Figure 4 shows the total time users spent on Facebook over years according to the comScore statistics. We see that the total time users spent on Facebook has been increasing quickly. It was reported that web users spend more time on Facebook than Google sites. On average, more than 8 billion minutes are spent on Facebook every day. As long as a user is online, his/her cached videos can be fetched for P2P video sharing. This makes a P2P video sharing system very suitable for Facebook since a fundamental prerequisite of P2P video sharing systems is that there are enough peers online to participate in video sharing.

O3: *Facebook users normally spend long time periods online, which makes P2P video sharing in Facebook feasible.*

2.2 Effect of Social Distance on Video Viewing Patterns

Social distance between two users in the social network graph represents the closeness of their relationship. If two users are directly connected in the social network, their social distance is 1; if one user is a friend of another user's friend, then the social distance between them is 2, and so on. Next, we investigate the impact of social distance on user video viewing patterns.

Among 52,500 video watching activities involving 12,000 users, we measured the social distance of a video viewer from the video owner, and show the distribution in Figure 5. We find that most of the viewers (around 70%) are 1-hop friends of the video owner, 2-hop viewers account for a portion of about 20%, and the remaining 10% of viewers watched videos of video owners are more than 2 hops away.

O4: *In Facebook, more than 90% of the viewers of a video are within 2 hops in the video owner's social network.*

We define a video viewer group as all users who have watched the video owner's videos. From O4, we obtain the inference (I):

I1: *A video viewer group of a video owner in Facebook is mostly within the 2-hop friend circle of the owner.*

Note that a user may own more than one video. To further identify the impact of social relationships on video viewing patterns, we selected the users who have multiple videos from our dataset and inspected the viewer group of each video owner. We classified the viewers of a video owner based on the percentage of the owner's videos they watched and calculated the distribution of different viewer classes in a viewer group. Figure 6 shows the average values of the percent versus the ratio of videos watched from the video owners. We observe that:

O5: *On average, in a user's viewer group, 25% of viewers watched all, 33% of viewers watched 80%, and all viewers watched 20% of the user's videos.*

We call the viewers who have watched almost all videos of a user the user's *followers*, and call other viewers *non-followers*. We use a threshold T_h for the percent of all the videos of a user that a viewer watches in order to become a follower, and set T_h=80% in this analysis. Figure 7 and Figure 8 show the distribution of followers and non-followers in terms of the social distance with the video owner. We can see from the figures that :

O6: *Viewers that watch almost all of a user's videos (i.e., followers) usually are 1-hop friends of the user, while most of other viewers (i.e., non-followers) are 1-hop or 2-hop friends of the user.*

2.3 Effect of Interest on Video Viewing Pattern

Next, we explore the correlation between user interests and video viewing patterns. We selected a sample of 118 distinct users that watched more than one video from our dataset and manually classified the videos they watched into 19 interest groups based on video content. The 19 interest groups were determined based on the video categories in YouTube such as gaming, rock music and action movie. For each user, we calculated the percentage of viewed videos of each interest group. Then, we ranked these 19 interest groups in descending order of the percentage values. We calculated the average percentage value of the 118 users for each interest group rank and show the result in Figure 9. We observe that, on average, 46% of videos a user watched are on his/her top 2 interests topics, 79% of videos a user watched are on his/her top 3 interests topics, and 94% are on his/her top 4 interests topics. The result implies that

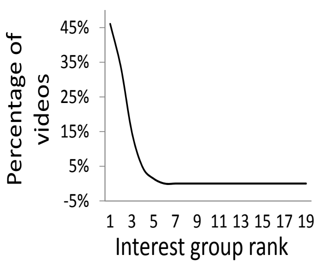


Fig. 9: Interest clustering effect

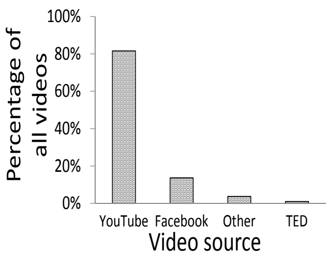


Fig. 10: Video source in Facebook.

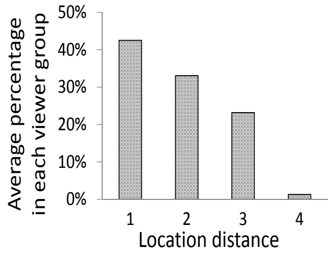


Fig. 11: Physical distance.

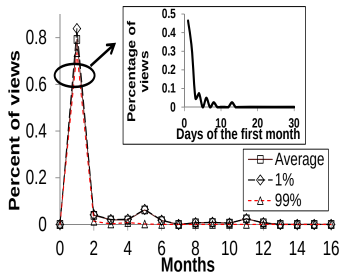


Fig. 12: Video active period.

the videos each user watches are generally orientated towards his/her few primary interests.

O7: *Users tend to watch the videos of their interests and each user generally has ≤ 4 video interests.*

A user can post on Facebook either self-uploaded videos or external video links from a third party video service provider such as YouTube. The video linking in Facebook is called "share", by which users can share links to videos they find interesting with their friends. Figure 10 shows the different sources of videos in our collected dataset. We see that the self-uploaded videos in Facebook account for about 14% of all videos. Others are external video links. YouTube, as the largest video site in the world, accounts for over 80% of all external links. Many other video sources such as TED and Hulu account for the remaining percentage.

For videos uploaded to Facebook, social relationships are the primary consideration when viewers decide whether to watch a video. For external videos, interest gains more weight in influencing the watching probability. However, no matter if it is a video uploaded to Facebook or an external link, we assume that the video owner, the video uploader, or the one who shared the external video link would have the video in his/her local cache. This assumption is reasonable because in the case of uploaded videos, the uploaders have the original videos, while in the case of external links, a user usually shares a video after (s)he has watched the video (or at least part of it) and hence has the video in cache. Therefore, both types of videos are applicable in P2P video sharing, in which videos are prefetched from users' local cache instead of the video server.

O8: *A large percentage of videos in Facebook are from YouTube, where the user video viewing patterns are driven by interests.*

Combining O4-O8, we can find that different watching incentives can be applied to different types of viewers. The followers of a user watch most of the user's videos regardless of the video content because of their close social relationship (e.g., close friends and fervent admirers). For the viewers that watch only a few of the user's videos, interest in the video content is a more important incentive. Additionally, some show a mixed video watching incentive. Thus, we infer

I2: *Followers are primarily driven by social relationship to watch videos, while non-followers are driven mainly by interest.*

2.4 Effect of Physical Location on Video Viewing Patterns

We also analyze the geographical locations of users who view the same videos in order to see whether location can also be leveraged for video sharing in OSNs. In Facebook, some users input their current resident city in their profiles. The location is in the format of "city, state" (e.g., Los Angeles, CA). For international users, the location is in the format of "city, country" (e.g., Tokyo, Japan). We denote two users located in the same city, state, or country as 1, 2 and 3, respectively, and two users located in different countries as 4. To investigate the location distribution of viewers, we calculated the percentage of viewers in each viewer group corresponding to different location distances between the viewer and video owner. We plot the average value of all viewer groups in Figure 11. From the figure, we can see that most users watching the same video are physically close to each other. Because many friend relationships in Facebook are connected by offline relationship, such as classmates or colleagues, this produces a strong location clustering effect. This result conforms to the observation in [24] that most of the wall posts are sent within local physical region. This effect could make P2P video sharing systems in OSNs more efficient by enabling geographically close nodes to share videos between each other.

O9: *Most users watching the same video are physically close, and on average about 40% of users watching the same video are in the same city.*

2.5 Active Life Period of Videos

We measured the percentage of views of a video in each month after the video is uploaded out of all views. Figure 12 shows the average of these values. We found that videos in Facebook have an active life period of about one month. Views in this period account for more than 90 percent of all views. After one month, there are only occasional views. The small figure inside Figure 12 more clearly shows the decreasing active life over days in the first month. We find that it follows a power-law distribution.

O10: *The videos in Facebook have an active life period of about one month on average. The decrease in the number of views of a video follows an exponential distribution.*

3 The Design of SocialTube

The Facebook measurement in Section 2 shows that video sharing in Facebook is increasingly popular (O1) and may generate a heavy burden on the video server (O2). Fortunately, the length of time that the users stay online in Facebook is rapidly increasing (O3), enabling the possibility for P2P video sharing among the online users themselves. Therefore, P2P-assisted video sharing is a promising strategy in OSNs. Based on observations O4 - O10 on the characteristics of video viewer behavior in Facebook, we propose SocialTube, a P2P video sharing system for OSNs.

We first introduce the basic concepts and strategies used in SocialTube. In Facebook, each node can upload a video to the Facebook video server or an external link to a video from an external server. In this paper, we use *server* to represent all video source servers, including both Facebook and external video servers. Similar to current peer-assisted content delivery mechanisms, the peers in SocialTube store videos they have watched previously for video re-distribution. In SocialTube, a video is divided into small chunks with a fixed size. Thus, a video viewer only needs to download the corresponding chunks of the video segment (s)he wants to watch.

3.1 Social Network based P2P Overlay Construction Algorithm

To identify followers and non-followers of a source node for structure construction, SocialTube pre-defines two thresholds, T_h and T_l, for the percent of videos in the source node that a viewer has watched during a time unit, say one week. If the percent value of a viewer is $\geq T_h$, the viewer is a follower. If the percent is $T_l < x \leq T_h$, the viewer is a non-follower.

Video sharing in Facebook distinguishes itself from other video sharing websites (e.g., YouTube) in two aspects: video sharing scope and video watching incentives. First, other websites provide system-wide video sharing where a user can watch any video, while in Facebook, videos are usually shared in a 2-hop small circle of friends (I1). Second, users in other video sharing websites are driven to watch videos by interests, while in Facebook, the followers of a source node (i.e., video owner) are driven to watch almost all of the source's videos primarily by social relationship, and non-followers watch a certain amount of videos mainly driven by interest (I2). According to these differentiating aspects, we design the P2P overlay structure, which is shown in Figure 13.

Based on I1, SocialTube establishes a *per-node* (in contrast to *per-video* in YouTube) P2P overlay for each source node. It consists of peers within 2 hops to the source that watch at least a certain percentage ($> T_l$) of the source's videos. Other peers can still fetch videos from the server. As shown in the figure, such peers of a source node S in the social network constitute a P2P overlay for the source node. We aim to achieve an optimal tradeoff between P2P overlay maintenance costs and video sharing efficiency. Some nodes within 2 hops may watch only a few videos in a source. Including these nodes and users beyond 2-hops into the overlay generates a greater structure maintenance cost than video sharing benefits. Based on I2, we build a hierarchical structure that connects a source node with its socially-close followers, and connects the followers with other non-followers. Thus, the followers can quickly receive chunks from the source node, and also function as a pseudo-source to distribute chunks to other friends. The source pushes the first chunk of its new video to its followers. The chunk is cached in each follower and has high probability of being used since followers watch almost all videos of the source. Further, non-followers sharing the same interest are grouped into an interest cluster for video sharing. We call peers in an interest cluster *interest-cluster-peers*. A node that has multiple interests is in multiple interest clusters of the source node.

Because the source node and followers are involved in every interest cluster for providing video content, we call

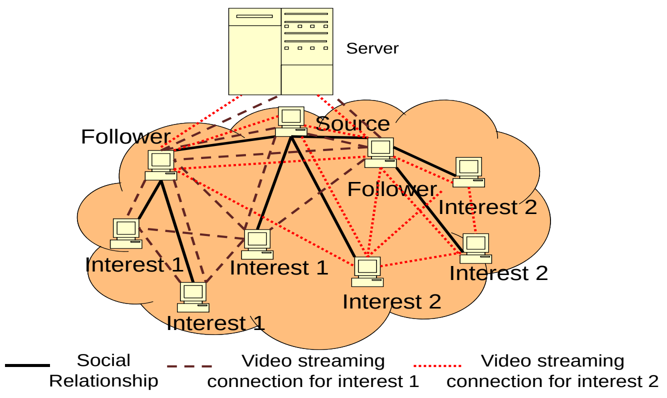


Fig. 13: Structure of SocialTube.

the group formed by the source, followers, and interest-cluster-peers in an interest cluster *swarm*, and call all nodes in a swarm *swarm-peers*. As I1 indicates, the cluster size of each interest cluster should be small. O9 indicates that many viewers of a video are physically close peers. Therefore, in order to reduce delay, physically close interest-cluster-peers are randomly connected with each other. The peers find their physically close peers based on their ISP, subnet information [25]. To preserve the privacy protection on OSN, we can add a constraint in which peer A can connect to peer B only when peer A is peer B's friend or can access peer B's shared videos. In Figure 13, the viewers of S form into two swarms. Because the nodes in each swarm have a high probability of owning chunks of the same video, they can retrieve chunks from their swarm-peers without relying on querying the server or large scale query flooding.

In current video sharing in Facebook, a node always requests the server for videos uploaded by source nodes. We let the server keep track of the video watching activities of viewers of a specific source node in order to identify and update its followers and non-followers based on SocialTube's pre-defined thresholds of T_l and T_h. This duty can be assigned to the source node itself if it has sufficient capacity. The nodes in the system will periodically report their video providing activities to the server. When the server determines that a peer is a follower of the source node, it notifies the source node, which notifies all nodes in its swarms about the follower. Consequently, the follower becomes a member of each of the swarms, and all swarm-peers in each of the swarms connect to it. When the server determines that a peer is a non-follower of the source node, the server determines its interests based on the contents of videos the peer visited, and notifies the source node about the non-follower along with its interests. The source node then notifies the peers in the clusters of the interests of that non-follower, and notifies the non-follower about the clusters. The non-follower connects to all followers and the source and to a few physically close nodes in each cluster. Consequently, the non-follower becomes a member of the swarm of each of the interest clusters. The server also periodically updates the roles of the followers and non-followers. If a node becomes neither follower nor non-follower, the server removes it by notifying others to disconnect from the node. If a follower becomes a non-follower, its connections are also updated accordingly. In an OSN, after a node logs out (i.e., leaves) the system, it will always logs in (i.e., joins in), so the SocialTube overlay does not update due to node departures. Only when the server notices that a node did not join in after a very long time departure, the node is removed from the overlay. Neighbors in the overlay periodically exchange messages. When a node notices that its neighbor is offline, it marks the connection and will unmark it when the neighbor is online. A node does not request videos from neighbors with marked connections. The nodes in a P2P structure, including the source, followers and non-followers, remember their roles and connections. Next time when a node goes online, it automatically connects to its previous neighbors and function based on its role.

As indicated in Figure 13, the source node has two followers, and its videos can be divided into two interest categories based on video content. The 1-hop and 2-hop friends of the source node with interest 1 and interest 2 form into two clusters, respectively. The source node and the followers are in each interest cluster, all of which form a swarm.

3.2 Social Network based Prefetching Algorithm

To reduce the video startup latency, we propose a push-based video prefetching mechanism in SocialTube. In SocialTube, when a source node uploads a new video to the server, it also pushes the prefix (i.e. first chunk) of the video to its followers and to the interest-cluster-peers in the interest clusters matching the content of the video. The prefix receivers store the prefix in their cache. Those interest-cluster-peers and followers who are not online when the source node pushes the prefix will automatically receive it from the source node or the server once they come online. After the source node leaves, the responsibility to push the prefix falls to the server. Since these followers and interest-cluster-peers are very likely to watch the video, the cached prefixes have a high probability of being used.

Once the nodes request the videos, the locally stored prefix can be played immediately without delay. Meanwhile, the node tries to retrieve the remaining video chunks from its swarm-peers. Similar to BitTorrent, SocialTube allows a requester to request 4 online nodes at the same time to provide the video content in order to guarantee provider availability and achieve low delay by retrieving chunks in parallel. It first contacts interest-cluster-peers, then followers, then the source node. If the requester still cannot find 4 providers after the source node is contacted, it resorts to the server as the only provider. Considering the high capacity of the server, the requester does not need to have 4 providers if it has the server as a provider. This querying order can distribute the load of chunk delivery among the swarm-peers while providing high chunk availability. The algorithm takes advantage of all resources for efficient video sharing without overloading specific nodes. The server can guarantee the availability of the video, even if the number of online users in a swarm is small.

4 Performance Evaluation

We evaluate the performance of SocialTube through both simulations on the event-driven simulator PeerSim [26] and PlanetLab [27] prototype implementation. We run each experiment for 10 times and report results within 95% confidential intervals.

TABLE 1: Bandwidth capacity and distribution of users

Groups	Downloading bandwidth	Uploading bandwidth	Percentage of nodes
1.	768k/s	128k/s	21.4%
2.	1536k/s	384k/s	23.3%
3.	3072k/s	768k/s	55.3%

4.1 Experiment Settings

Our simulation setup is based on a part of our crawled dataset, which contains approximately 2,000 distinct videos. We selected 5,000 nodes from the trace data. The link delay for nodes with location distance x as defined in Figure 11 was set to x seconds. The connectivity degree of these selected nodes follows power-law distribution with skew factor equals to 0.5 [28]. We assigned each of the 2,000 videos to a randomly chosen node in the system. To simulate the real Internet environment, where peers have heterogenous bandwidth, we used the statistics in Table 1 used in [9]. To simulate the geographic locations of nodes, each node has a location ID in [1-10]. The nodes with numerically closer ID are geographically closer nodes. We used the distribution of friendship in spatial proximity from [29] in our experiments.

Table 2 shows the default parameters unless otherwise specified. The video bitrate was set to 330 kbps. Based on empirical observations, we assume that whenever a node cannot receive a chunk in time for viewing, it pauses for 3 seconds and then resumes playback. The file size of a video was randomly chosen from [20-30] MB, which is a normal size for a video with a length of 2-3 minutes.

Since every person spends about 420 minutes on Facebook per month [30] on average, we can infer that every person spends about 14 minutes and hence watches at most 4 videos per day on average. We used simulation cycle to simulate one day. In each simulation cycle, the number of videos a peer watched was randomly chosen from [1-4]. one simulation lasted for 100 simulation cycles. Based on O7, each node in the system randomly selected [2-4] interest categories as its interests.

Based on O6, every source node randomly selected 20% of its friends as its followers. As the practical user video viewing behaviors in Facebook, every node can see the names of the videos owned or shared by its friends. Based on O6 and O7, unless otherwise indicated, from these videos, a node randomly selected videos in its interests or its followees to view. Recall that a node can access videos either owned or shared by its friends in Facebook. In practice, after a node watches a video, it may or may not share the video. To simulate this behavior, each node first randomly selected a number $x \in [1, 40]$ and shared its watched x^{th} video, and repeated this process until the simulation completed.

We compare the performance of SocialTube with other representative works in peer-assisted video streaming, PA-VoD [8], NetTube [9] and Random (as a baseline). In PA-VoD, physically close peers with the same location ID are clustered for video sharing between each other. In NetTube, peers that have similar interests are clustered together for video sharing. The details of PA-VoD and NetTube are presented in Section 9. Random clusters peers randomly into 10 groups regardless of their friendships, locations and interests. Nodes in a cluster connect with each other randomly, and use flooding for video search. In each simulation cycle, a source node pushes one video prefix to its followers and the interest-cluster-peers in the interest cluster matching the video in SocialTube; the server pushes the prefix of one video to the nodes in the interest cluster of the video in NetTube, and to the nodes in the cluster of the video's uploader in PA-VoD and Random. After a node selects a video to view, it searches its interest cluster in NetTube, and searches its own cluster and a neighboring cluster in PA-VoD and Random. It finds one video provider to fetch video chunks; if it cannot find or access the requested video, it resorts to the server. To make the methods comparable, SocialTube does not include its chunk delivery and scheduling algorithm and buffer management unless otherwise indicated.

TABLE 2: Experiment default parameters.

Parameter	Default value
Number of clients	5,000
Number of videos	2,000
Number of interest categories	19
Number of interests per client	2-4
Trace duration	40 days
Chunk size	3 MBytes
Prefix length	3 MBytes
Server uploading bandwidth	20Mbps
Video size	Distribution of YouTube videos
Cache size	300MBytes

We used the following metrics in the experiments:

• *Prefetching accuracy.* This is the probability that a user requests a video whose prefix is in its cache and it can access the prefix's video. This metric shows the effectiveness of the prefetching and enhancing the video playback continuousness.

• *Chunk transmission delay.* This is the chunk transmission time between two peers. This metric shows the delay in retrieving video chunks.

• *The number of searched clients for a video.* This is the number of unique nodes that are queried before a node finds a video provider or fails to find a video from clients. This metric shows the efficiency in video provider searching.

• *Percent of server contribution.* This is the ratio of server bandwidth consumed in a system over the total bandwidth consumed in the client/server system. The bandwidth is measured by the total size of served videos. This metric shows the effectiveness of reducing the bandwidth burden of the server.

• *Playback continuousness probability.* This is the percent of nodes that have not experienced freezing caused by waiting for a missing chunk in video playback due to a user's forward skipping when watching video.

• *Startup delay.* This is the time elapsed after a node selects a video and before the video starts to play. This metric reflects the effectiveness of a prefetching mechanism.

• *Buffering delay.* This is the total time for a user to receive a certain number of chunks after sending out a video request.

• *Average overlay maintenance cost.* This is the number of communication messages between neighboring nodes for overlay maintenance.

4.2 Effectiveness of the Prefetching Algorithm

In this experiment, we let each node randomly request videos whose prefixes are in its cache. A node cannot access a video not owned or shared by its friends. We varied the number of nodes in the system from 1,000 to 5,000 with 500 increase in each step. Figure 14 shows the average prefetching accuracy versus client (i.e., node) population. We see that the prefetching accuracy of SocialTube remains higher than 0.9, those of PA-VoD and Random remain at 0.03 and nearly 0, respectively, and that of NetTube lies in the middle. Also, as the client population increases, unlike other methods, the prefetching accuracy of NetTube decreases significantly. SocialTube explores the social relationship and clusters the follower and similar-interest peers within 2-hops from a source together. A source node pushes its video prefixes to its followers and the cluster nodes of the video's interest. In a source's per-node overlay, nodes 2 hops away from the source cannot access its videos unless the videos are shared by their friends, and 1-hop friends of the source can always access the videos. Thus, SocialTube achieves high prefetching accuracy but it is not 100%. As the client population increases, the size of interest cluster increases. Then, more cluster-peers within 2-hops of the source can visit shared videos from their friends, thus the video prefetching accuracy remains nearly constant. NetTube only clusters peers with similar interests without considering the social relationship. When a video prefix is pushed to the cluster of the video's interest, some nodes are unable to access the video since they are not the friends of the source or the video sharers. Given a certain number of videos in the system, as the client population increases, the number of videos in a node decreases and then the number of accessible videos from friends of a node decreases, leading to a decreasing prefetching accuracy. PA-VoD clusters physically close nodes without considering interests or social relationship. A node may not be in the same cluster with its friends or the video sharers, leading to low prefetching accuracy. In Random, nodes are randomly clustered. Thus, nodes in a cluster receiving a prefix of a video are unlikely to be the friends of the source node or the video sharers the video, leading to very low prefetching accuracy.

In this experiment, we assigned 6 videos to each source node. We randomly selected 5 per-node overlays and chose a client in each overlay. We recorded each node's prefetching accuracy when it has x ($x = 1, 2 \cdots, 5$) video prefixes from the source node. Figure 15 shows the average prefetching accuracy of the 5 nodes versus the number of prefetched videos of each node. As the number of prefetched videos increases, the prefetching accuracy of SocialTube, NetTube and PA-VoD increases. This is because as a peer receives more video prefixes, when it chooses a video to view, it has a higher probability to have the prefix of the chosen video in its cache. We also see that NetTube leads to lower prefetching accuracy than SocialTube. Also, PA-VoD and Random lead to extremely low prefetching accuracy. In SocialTube, a source pushes its video prefix to its followers and the interest-cluster of the video's interests in its per-node overlay. In NetTube, a video prefix is pushed to all nodes that share the same interest of the video's interest. Thus, when a follower chooses a video in its followee to view, the prefix of this video is in its cache in SocialTube but may not be in its cache in NetTbe. In PA-VoD, a video prefix is pushed to the physically close nodes of the video uploader, which have low probability to be the source's friends or the video sharers, thus generating very low prefetching accuracy. We also find

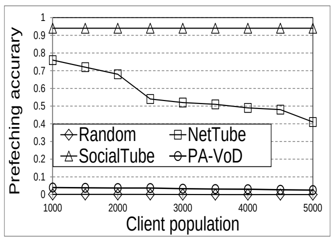


Fig. 14: Prefetching accuracy vs. node population.

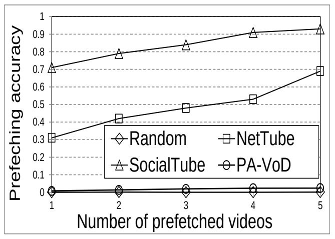


Fig. 15: Prefetching accuracy vs. # of prefetched videos.

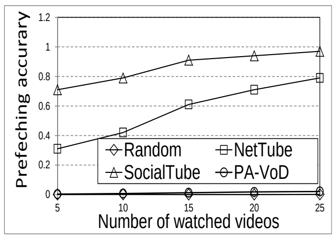


Fig. 16: Prefetching accuracy vs. # of watched videos.

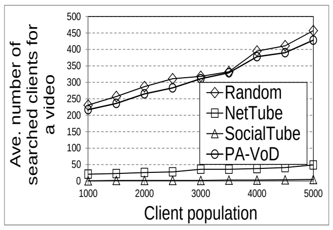


Fig. 17: # of searched clients vs. client population.

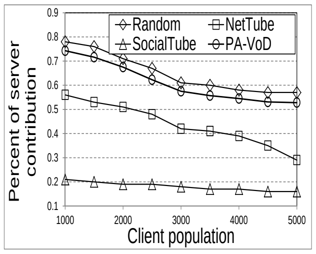


Fig. 18: Percent of server contribution vs. client population.

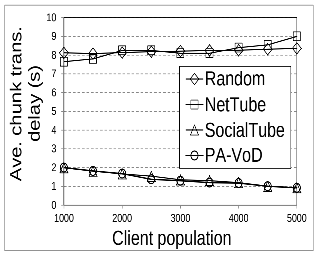


Fig. 19: Chunk transmission delay vs. client population.

that the prefetching accuracy in Random maintains at around 0 due to the same reason explained previously.

In this experiment, we randomly selected 5 per-node overlays and chose a client in each overlay. We let each source node have 30 videos in 5 interests. A client initially has [2-4] interests. In every simulation cycle, each client receives a prefix from its source node and chooses [1-4] videos to watch. If the client chooses a video not in its current interests, the video's interest is added to the client's interest list and it joins in the interest-cluster of this interest and receives the pushed prefix of videos in this interest in SocialTube and NetTube. Figure 16 shows the average value of the prefetching accuracies of the 5 clients versus the number of watched videos of a client over time. The figure illustrates that as the number of watched videos increases, the prefetching accuracy of SocialTube and NetTube increases. This is because in SocialTube and NetTube, as a client watches more videos, it joins in more interest-clusters and receives more prefixes, and thus its prefetching accuracy increases. Again, NetTube generates lower prefetching accuracy than SocialTube due to the same reason as in Figure 15. The prefetching accuracy of PA-VoD and Random is not affected by the number of watched videos as they do not have interest clusters. Also, they produce low prefetching accuracy due to the same reasons as explained before.

4.3 Scalability Performance

Figure 17 shows the average number of searched clients for a video provider versus the client population. We see that Random and PA-VoD need to search significantly more clients than NetTube and SocialTube. In Random, as the cluster nodes are unlikely to have the same interests or friend relationship, the viewers need to search much more nodes for the requested videos. In PA-VoD, cluster nodes are physically close nodes that may have friendship, thus it has fewer searched clients for a video than Random. In contrast, in SocialTube and NetTube, the nodes that are likely to watch the same video are clustered. Therefore, they have high probability to find the requested video in their cluster. We also find that NetTube generates more searched clients than SocialTube. This is because in NetTube, nodes in one interest cluster may not have the permission to access videos in some of the nodes in the cluster due to the constraint in OSNs. Also, when a node visits its followee's video, the followee may not be in the interest cluster of the video. In SocialTube, the common-interest nodes within two hops of a source node and its followers are clustered. Thus, a node has a high probability to find and access the requested video from a cluster node. The figure also shows that as the client population increases, the video search overhead increases more rapidly in Random and PA-VoD than in NetTube and SocialTube. A larger client population increases the cluster size, so a node needs to search more cluster nodes in Random and PA-VoD. We notice a slight increase in the search overhead in NetTube, because when more nodes are included in a cluster in NetTube, the probability that a node can find a friend decreases. In SocialTube, as only the common-interest nodes within 2-hops and followers of a source node are clustered, the client population increase does not greatly increase a cluster size, so both the video access probability and provider location probability remain approximately the same.

In the following experiments in this section, we let each requester find four video providers to fetch video chunks at the same time. Figure 18 illustrates the percent of server contribution versus client population. It shows that as the client population increases, the percent of server contribution in all systems decreases. Higher client population increases the number of friends of a node that can provide services. It also causes more clients viewing videos, leading to faster video dissemination to a wider friend circle around a video owner. Then nodes have higher probability of finding videos from peers, thus reducing the bandwidth consumption of the server. We also see that the percent of server contribution follows SocialTube<NetTube<PA-VoD<Random. This is because the nodes in SocialTube can locate the video chunks from other peers more efficiently, since it considers both social relationship and interest. Common-interest nodes are more likely to be friends than physically close nodes, so NetTube can locate video providers more efficiently than PA-VoD. Randomly clustered nodes in Random are unlikely to be friends, so Random produces the lowest video provider location efficiency.

Figure 19 plots the average chunk transmission delay versus client population. We did not consider the transmission from the central server in calculating this metric. We see that SocialTube and PA-VoD have almost the same chunk transmission delay, which is dramat-

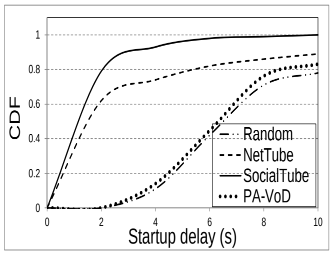


Fig. 20: CDF of node startup delay.

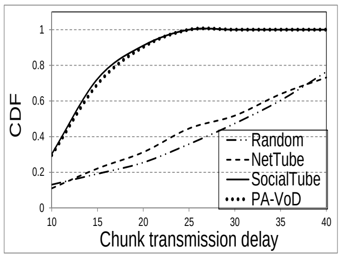


Fig. 21: CDF of chunk transmission delay.

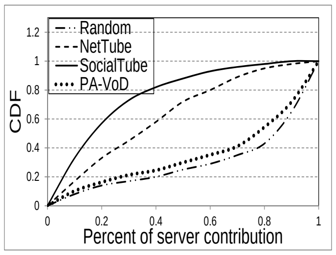


Fig. 22: CDF of the percent of server contribution in different systems.

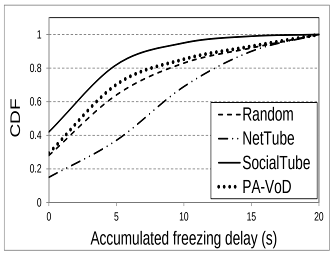


Fig. 23: CDF of accumulated freezing delay.

ically lower than those of NetTube and Random. In SocialTube and PA-VoD, the nodes contact the physically close nodes for the video chunks. NetTube and Random do not consider the physical locations of nodes. We also see that as the client population grows, the chunk transmission delay of SocialTube and PA-VoD decreases slightly, while that of NetTube and Random exhibit a marginal increase. As client population increases, a video requester has a higher probability to find physically nearby video holder in SocialTube and PA-VoD, while the average distance between video requester and video provider increases in NetTube and Random.

5 Experimental results on PlanetLab

We implemented SocialTube based on the Facebook video trace on the PlanetLab real-world testbed [27]. We randomly chose 268 online nodes in the PlanetLab and installed the server at 128.112.139.26 located in Princeton University. We used a smaller set of 50 distinct videos from our dataset for testing.

Figure 20 shows the CDF of the average startup delay. It shows that about 80% of the nodes in SocialTube and 65% of the nodes in NetTube have a startup delay of less than 2s, but the startup delay of all nodes in PA-VoD and Random is larger than 2s and the startup delay of 20% of their nodes is even larger than 10s. This is mainly due to the high effectiveness of the prefix prefetching mechanism in SocialTube and NetTube and its low effectiveness in PA-VoD and Random as explained previously.

Figure 21 shows the CDF of the average chunk transmission delay between nodes. We find that NetTube and Random generate much higher delay than SocialTube and PA-VoD. Since NetTube and Random do not consider the node geographic locations, a node may contact geographically distant nodes for video chunks, which increases the chunk transmission delay. Both SocialTube and PA-VoD try to fetch chunks from geographically close nodes, thus their chunk delivery delays remain low and are almost the same.

Figure 22 shows the CDF of the percent of server contribution for video transmission. We see that about 90% of the nodes in SocialTube and 80% of the nodes in NetTube have no more than 0.6 server contribution. Around 60% of the nodes have more than 0.7 and 0.78 server contribution in PA-VoD and Random, respectively. Since SocialTube considers social relationship and interests in node clustering, the probability that nodes find video providers is high. Therefore, SocialTube has much less bandwidth demand from server than other methods. NetTube clusters the nodes with the same interests. As followers may not find requested videos of their followees in their interest clusters in NetTube, so it produces higher server contribution than SocialTube. PA-VoD clusters the physically close nodes that may not be friends, leading to lower probability of finding video providers hence higher server contribution than NetTube. Random only randomly clusters nodes, so it incurs the highest server bandwidth contribution. The result shows that SocialTube is most effective in reducing the bandwidth burden on the central video server.

Figure 23 shows the CDF of the accumulated freezing delay, which is defined as the total freezing delay when a video is playing. We see that the percentage of the users that did not experience any freezing delay in SocialTube, PA-VoD, Random and NetTube is 42%, 30%, 28% and 15%, respectively. Also, 80% of the users experienced accumulated freezing delay less than 5s, 7s, 9s and 12s in SocialTube, PA-VoD, Random and NetTube, respectively. That is, the users experience more video freezing delay in NetTube than in PA-VoD and Random, and experience the smallest freezing delay in SocialTube. In SocialTube, the video chunks can be more accurately prefetched since nodes push video chunks to their friends that are very likely to watch the videos. The interest-based prefetch mechanism in NetTube and locality-based prefetch mechanism in PA-VoD are not as accurate as SocialTube. In addition, in SocialTube, when a user watches the first chunk of the video, the other video chunks are transmitted from other users that are physically close to the user. Therefore, the nodes are unlikely to experience playback freezing. In PA-VoD, nodes receive chunks from either physically close nodes or the server, both of which provide quick transmission. Therefore, PA-VoD produces the second smallest freezing time. In Random, nodes receive chunks mainly from the server, so chunk transmission delay is relatively low. As NetTube can locate video providers with high probability but video providers are not physically close to video requesters, the users in NetTube experience the longest freezing delay.

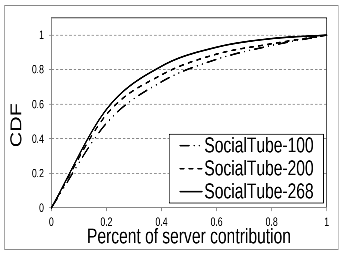


Fig. 24: CDF of the percent of server contribution in SocialTube.

Figure 24 shows the CDF of the percent of server contribution of each node with different client population in SocialTube. It shows that in the system with 268 nodes, 80% of the nodes require no more than 40% of the traffic from the server, and in the system with 200 and 100 nodes respectively, 80% of the nodes require no more than 50% and 56% of the traffic from the server, respectively. As the system has more nodes, more nodes can share videos after they watch the videos. Then, the total uploading capacity of the P2P transmission increases, which reduces the burden on the server. This result implies that P2P structure is an optimal architecture for a large-scale video sharing system.

6 Conclusion

Video sharing is an increasingly popular application in OSNs. However, the client/server architecture deployed by current video sharing systems in OSNs costs a large amount of resources (i.e. money, server storage) for the service provider and lacks scalability. Meanwhile, because of the privacy constraints in OSNs, the current peer-assisted Video-on-Demand (VoD) techniques are suboptimal if not entirely applicable to the video sharing in OSNs. In this paper, we crawled video watching trace data in one of the largest online social network websites Facebook, from Jul. 2007 to Aug. 2010 and explored the users' video viewing patterns. We found that in a user's viewer group, 25% viewers watched all videos of the user driven by social relationship, and the viewing pattern of the remaining nodes is driven by interest. Based on the observed social and interest relationship in video watching activities, we propose SocialTube, which provides efficient P2P-assisted video sharing services. Extensive simulation results show that SocialTube can provide a low video startup delay and low server traffic demand. We also implemented a prototype in PlanetLab to evaluate the performance of SocialTube. The experimental results from the prototype further confirm the efficiency of SocialTube.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and U.S. Department of Energy's Oak Ridge National Laboratory including the Extreme Scale Systems Center located at ORNL and DoD 4000111689. An early version of this work was presented in the Proceedings of Infocom-mini'12 [22].

References

[1] Facebook passes google in time spent on site for first time ever. http://www.businessinsider.com/.

[2] Social media, web 2.0 and internet stats. http://thefuturebuzz.com/2009/01/12/social-media-web-20-internet-numbers-stats/.

[3] K. Wang and C. Lin. Insight into the P2P-VoD system: Performance modeling and analysis. In *Proc. of ICCCN*, 2009.

[4] Y. Huang, Z. Fu, D. Chiu, C. Lui, and C. Huang. Challenges, design and analysis of a large-scale P2P VoD system. In *Proc. SIGCOMM*, 2008.

[5] B. Cheng, L. Stein, H. Jin, X. Liao, and Z. Zhang. Gridcast: improving peer sharing for p2p vod. *ACM TOMCCAP*, 2008.

[6] C.-P. Ho, S.-Y. Lee, and J.-Y. Yu. Cluster-based replication for P2P-based video-on-demand service. In *Proc. of ICEIE*, 2010.

[7] J. Wang, C. Huang, and J. Li. On ISP-friendly rate allocation for peer-assisted VoD. In *Proc. of MM*, 2008.

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

[9] X. Cheng and J. Liu. NetTube: Exploring social networks for peer-to-peer short video sharing. In *Proc. of INFOCOM*, 2009.

[10] B. Li, M. Ma, Z. Jin, and D. Zhao. Investigation of a large-scale P2P VoD overlay network by measurements. *Peer-to-Peer Networking and Applications*, 5(4):398–411, 2012.

[11] C. Wu, Z. Li, X. Qiu, and F. Lau. Auction-based P2P VoD streaming: Incentives and optimal scheduling. *TOMCCAP*, 2012.

[12] T. Fujimoto, R. Endo, and H. Shigeno. P2P video-on-demand streaming using caching and reservation scheme based on video popularity. *IJGUC*, 3(2/3):188–199, 2012.

[13] W. Wu and J. Lui. Exploring the Optimal Replication Strategy in P2P-VoD Systems: Characterization and Evaluation. *TPDS*, 23(8):1492–1503, 2012.

[14] Y. Zhou, T. Fu, and D. Chiu. A unifying model and analysis of P2P VoD replication and scheduling. In *Proc. of INFOCOM*, pages 1530–1538, 2012.

[15] D. Niu, Z. Liu, B. Li, and S. Zhao. Demand forecast and performance prediction in peer-assisted on-demand streaming systems. In *Proc. of INFOCOM*, pages 421–425, 2011.

[16] L. Chang and J. Pan. Towards the optimal caching strategies of peer-assisted VoD systems with HD channels. In *Proc. of ICNP*, 2012.

[17] Z. Wang, L. Sun, S. Yang, and W. Zhu. Prefetching Strategy in Peer-Assisted Social Video Streaming. In *Proc. of MM*, 2011.

[18] A. Barrios, M. Barrios, D. Vera, P. Bocca, and C. Rostagnol. GoalBit: A Free and Open Source Peer-to-Peer Streaming Network. In *Proc. of ACM Multimedia*, 2011.

[19] W. P. K. Yiu, X. Jin, and S. H. G. Chan. VMesh: Distributed Segment Storage for Peer-to-peer Interactive Video Streaming. *IEEE JSAC*, 2007.

[20] H. Shen, L. Zhao, Z. Li, and J. Li. A DHT-aided chunk-driven overlay for scalable and efficient P2P live streaming. In *Proc. of ICPP*, 2010.

[21] H. Shen, L. Zhao, H. Chandler, J. Stokes, and J. Li. P2P-based multimedia sharing in user generated contents. In *Proc. of INFOCOM*, 2011.

[22] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li. SocialTube: P2P-assisted Video Sharing in Online Social Networks. In *Proc. of Infocom Mini Conference*, 2012.

[23] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proc. of INFOCOM*, 2010.

[24] M. Wittie, V. Pejovic, L. Deek, K. Almeroth, and B. Zhao. Exploiting locality of interest in online social networks. In *Proc. of Co-NEXT*, 2010.

[25] H. Wang, J. Liu, and K. Xu. On the locality of bittorrent-based video file swarming. In *Proc. of P2P*, 2009.

[26] Peersim: A peer-to-peer simulator. http://peersim.sourceforge.net/.

[27] PlanetLab. http://www.planet-lab.org/.

[28] A. Nazir, S. Raza, and C. Chuah. Unveiling dacebook: a measurement study of social network based applications. In *Proc. of SIGCOMM*, 2008.

[29] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proc. of WWW*, pages 61–70, 2010.

[30] Facebook users average 7 hrs a month in january as digital universe expands. http://blog.nielsen.com/nielsenwire.

[31] L. Zhou, Y. Zhang, K. Song, W. Jing, and V. Vasilakos. Distributed media services in p2p-based vehicular networks. *IEEE TVT*, 2011.

[32] P. Salvador and A. Nogueira. Study on geographical distribution and availability of bittorrent peers sharing video files. In *Proc. of ISCE*, 2008.

[33] T. Nair and P. Jayarekha. Strategic prefetching of vod programs based on art2 driven request clustering. *International Journal of Information*, 2011.

[34] D. Wu, C. Liang, Y. Liu, and K. Ross. View-upload decoupling: A redesign of multi-channel P2P video systems. In *Proc. of INFOCOM*, 2009.

[35] T. Silva, J. Almeida, and D. Guedes. Analysis of P2P streaming based on the characterization of live user-generated video. In *Proc. of ICC*, 2010.

[36] C. Wu, B. Li, and S. Zhao. Multi-channel live P2P streaming: Refocusing on servers. In *Proc. of INFOCOM*, 2008.

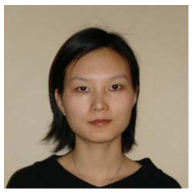
[37] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proc. of WWW*, 2007.

[38] J Liu X. Cheng C. Dale. Statistics and social network of YouTube videos. In *Proc. of IEEE IWQoS*, 2008.

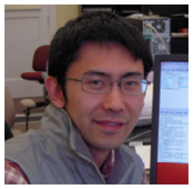
[39] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. *SIGOPS Oper. Syst. Rev.*, 2006.

[40] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. R. Van, and H. J. Sips. Tribler: a social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 2008.

[41] D. N. Kalofonos and Z. Antoniou. A hybrid p2p/infrastructure platform for personal and social internet services. In *Proc. of PIMRC*, 2008.



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



Ze Li Ze Li received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2007, and the Ph.D. degree in Computer Engineering from Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks. He is currently a data scientist in the MicroStrategy Incorporation.

Yuhua Lin Yuhua Lin received both his BS degree and MS degree in Computer Engineering from Sun Yat-sen University, China in 2009 and 2012 respectively. He is currently a Ph.D student in the Department of Electrical and Computer Engineering of Clemson University. His research interests include social networks and reputation systems.

Jin Li Dr. Jin Li is a Research Manager and Principal Researcher at Microsoft Research (Redmond, WA). He manages the Compression, Communication and Storage group. Blending theory and system, Dr. Li excels at interdisciplinary research, and is dedicated to advance communication and information theory and apply it to practical system building. He received his Ph.D. (with honor) from Tsinghua University in 1994. After brief stints at USC and Sharp Labs, he joined Microsoft Research in 1999, first as one of the founding members of Microsoft Research Asia, and then moved to Microsoft Research (Redmond, WA) in 2001. From 2000, Dr. Li has also served as an Affiliated Professor in Tsinghua University. Dr. Li's invention has been integrated into many Microsoft products. Recently, he and his group members have made key contributions to Microsoft product line (e.g., RemoteFX for WAN in Windows 8, Primary Data Deduplication in Windows Server 2012, and Local Reconstruction Coding in Windows Azure Storage), that leads to commercial impact in the order of hundreds of millions of dollars. Dr. Li was the recipient of Young Investigator Award from Visual Communication and Image Processing'98 (VCIP) in 1998, ICME 2009 Best Paper Award and USENIX ATC 2012 Best Paper Award. He is/was the Associate Editor/Guest Editor of IEEE Trans. On Multimedia, Journal of Selected Area of Communication, Journal of Visual Communication and Image Representation, P2P networking and applications, Journal of Communications. He was the General Chair of PV2009, the Lead Program Chair of ICME 2011, and the Technical Program Chair of CCNC 2013. He is an IEEE Fellow.

7 Enhanced Methods

7.1 Chunk Delivery and Scheduling

After discovering 4 providers, the requester downloads chunks from them until all chunks of the requested video are downloaded. Figure 25 shows the chunk delivery algorithm in SocialTube. At first, the node sends requests to providers to check whether they have the chunks it needs. All providers that have the required chunks send back their current available uploading bandwidth and chunk bitmap to the requesting node. Then, the requester can know how many chunks and which chunks each uploading node can provide. In order to guarantee a low streaming delay, we propose a policy for a downloading node to determine the number of chunks it should request from each uploading node.

Policy 1: The number of chunks a node can request from each provider, denoted by m_i, is determined based on the available uploading bandwidth of each provider. That is,

$$m_i = \frac{w_i}{\sum_{i=0}^{n_r} w_i} \cdot m, \tag{1}$$

where m is the total number of missing chunks of the requester, w_i is the uploading bandwidth of provider i, and n_r is the number of responding providers. The exact number of chunks required from a provider also depends on useful chunks the provider can provide. Because a requester can receive chunks from several providers simultaneously, the transmission delay of video streaming is reduced. Since the number of chunks assigned to providers is proportional to their available uploading bandwidth, and the node with higher uploading bandwidth should deliver more chunks, the uploading bandwidth is efficiently used to minimize the transmission delay. Once a video requester finds a chunk provider, it continuously downloads chunks from the provider until all requested chunks are retrieved or they are disconnected. In the latter case, the requester continues to find another provider.

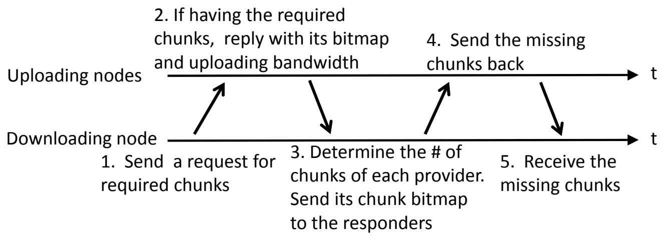

2. If having the required chunks, reply with its bitmap and uploading bandwidth
4. Send the missing chunks back
Uploading nodes
t
Downloading node
t
1. Send a request for required chunks
3. Determine the # of chunks of each provider. Send its chunk bitmap to the responders
5. Receive the missing chunks

Fig. 25: Chunk delivery algorithm.

A chunk provider may receive a number of chunk requests from different requesters. Thus, another question is how can a provider determine the priority to serve those requesters so that every requester's playback continuousness is guaranteed.

In SocialTube, each node uses a chunk window to measure how many chunks it has prefetched after the timestamp of current playback time. As shown in Figure 26, we use t_p to denote the current playback time of the video and t_f to denote the timestamp of the longest continuous chunk series. Then, the chunk window size is calculated by $W = t_f - t_p$. Because the t_f and t_p values of different requesters of one video provider are different respectively, their window sizes are different. The chunk window size represents the urgency of playback needs. In order to guarantee the continuousness of playback of all nodes, we propose another policy for uploading nodes to decide the priority of serving requesters.

Policy 2: For a number of chunk requests from different downloading nodes, the uploading node gives higher priority to the requester that has smallest chunk window size.

Figure 26 gives an explanation of Policy 2. After uploading peer C receives requests from peer A and peer B, it compares their chunk window sizes. Since the chunk size of peer A is larger than peer B, peer C uploads the chunks to peer B first, so that B will not experience a freeze due to the lack of prefetched chunks.

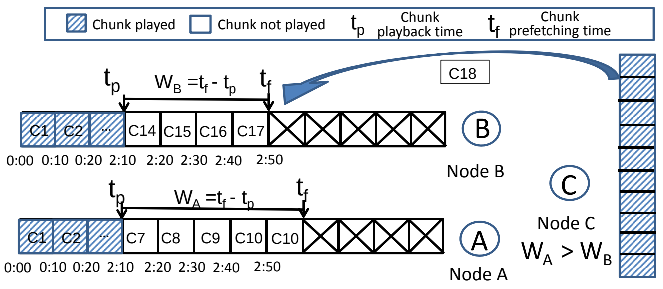


Fig. 26: Chunk scheduling algorithm.

7.2 Buffer Management Algorithm

As O10 shows, the frequency that a video is watched decreases over time. After a video has been published for one month, it is seldom watched by users. Therefore, nodes in SocialTube do not need to have a large cache to store all the videos that were watched before. According to O10, the decreasing number of views of a video can be modeled with a exponential distribution as $V = V_{max} \cdot a^{-t}$ $(a > 1)$, where V_{max} is the maximum number of views of the video, and t is the current life time of the video. Meanwhile, as videos have different popularities, videos with lower popularities should be replaced first, since these videos are less likely to be requested by other nodes. To calculate the popularity of a video, each node ranks the videos it holds based on the number of times the video has been viewed during a time unit. The video with rank 1 has a higher viewing frequency than the video with rank 2. Then, the probability that a video with rank i is queried by other nodes equals the percent of its views in the total views; that is, $p = \frac{1}{i} / \sum_{i=1}^{n} \frac{1}{i} = 1/(i \cdot \sum_{i=1}^{n} \frac{1}{i})$, where n is the number of videos in the node's cache and the videos are ranked from 1 through n [9]. Therefore, considering a video's active life time in Facebook and its popularity, we define the buffer replacement metric as

$$B = V \cdot p = \frac{V_{max} \cdot a^{-t}}{i \sum_{i=1}^{n} 1/i}. \tag{2}$$

The videos with large B have a higher probability to be watched, and the video with a small B should be replaced first.

8 Experimental Results

8.1 Effectiveness of the Chunk Delivery and Scheduling Algorithm

We then evaluate the two policies introduced in Section 7.1 that improve SocialTube. We use SocialTube-1-2 to denote SocialTube with both Policy 1 and Pol-

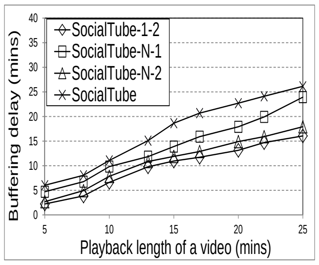


Fig. 27: Video buffering delay vs. playback length of a video.

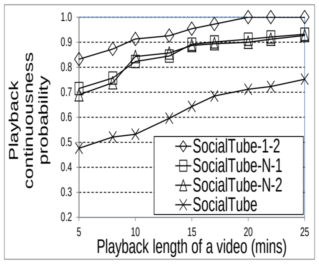


Fig. 28: Playback continuousness probability vs. playback length of a video.

icy 2, use SocialTube-N-1 and SocialTube-N-2 to denote SocialTube-1-2 without Policy 1 and Policy 2, respectively. In SocialTube-N-1, the chunk transmission loads are equally assigned to the uploading neighbors, and in SocialTube-N-2, a video provider sends video chunks to requesters based on the policy of "first come first serve". We use SocialTube to denote SocialTube without either policy 1 or policy 2. Figure 27 shows the average buffering delay versus the playback length of a video in minutes. We see that the buffering delay follows SocialTube>SocialTube-N-1>SocialTube-N-2>SocialTube-1-2. Based on Policy 1, SocialTube-1-2 can adaptively assign more chunk loads to the nodes with higher uploading bandwidths to reduce the video buffering delay. Therefore, SocialTube-1-2 generates much less video buffering delay than SocialTube-N-1. Based on Policy 2, SocialTube-1-2 gives higher priority to the users that have smaller playback window sizes when sending chunks to video receivers, thus preventing users from suffering from video playback freezes. Therefore, SocialTube-1-2 produces less buffering delay than SocialTube-N-2. As the delay due to the imbalanced chunk transmission load is larger than the delay due to chunk freezing when the nodes cannot receive chunks in time, the buffering delay in SocialTube-N-1 is larger than that in SocialTube-N-2. Without any enhanced policy, SocialTube produces the longest buffering delay. As the playback length of a video increases, the buffering delay of all methods increases since more chunks need more time to transmit.

In this experiment, each video viewer had a forwarding skipping when the playback length of a video equals x minutes, which was varied from 5 to 25 with 5 increase in each step. The skipping time was set to 10s, which approximately is the latency for transmitting one chunk. Figure 28 shows the playback continuousness probability. We see that for all these four methods, the playback continuousness probability increases as the playback length of a video increases. During the video playback time, the remaining chunks of the video are also fetched simultaneously. The longer a video plays, the more chunks that can be fetched, leading to higher playback continuousness probability. We also see that the playback continuousness probability follows SocialTube-1-2>SocialTube-N-1≈SocialTube-N-2>SocialTube. Recall that Policy 1 can reduce the average transmission delay between the nodes in the system. Therefore, the chunks can be fetched to the client's local buffer quickly. Policy 2 can reduce the probability of playback freeze. Therefore, the playback continuousness probability is high in SocialTube-1-2. Since the nodes do not use policy 1 in SocialTube-N-1, then the bandwidth utilization of some peers is low, which increases the average chunk fetching delay in the system. Since the nodes do not use Policy 2 in SocialTube-N-2, some nodes may quickly fetch many chunks while others may always experience playback freeze, leading to low playback continuousness probability. Therefore, only when both of two policies are concurrently used as in SocialTube-1-2, the playback continuousness of the nodes can be optimized. These results confirm the effectiveness of the two policies in improving the performance of SocialTube.

8.2 Effectiveness of the Buffer Management Algorithm

In this experiment, we let each node generate one video per simulation cycle. A node does not request videos generated 30 simulation cycles ago since the popularity of a video is about one month (O10). A node deletes its old videos based on our proposed buffer management algorithm if its cache is full. We varied the cache size of each node from 50M to 400M with an increase step of 50M. Figure 29 shows the average percent of server contribution of all video requests versus the cache size of each node. We see that the percent of server contribution follows SocialTube<NetTube<PA-VoD<Random due to the same reason as in Figure 18. We also see that the percent of server contribution of all three methods decreases as the cache size increases till 300M. A larger cache size increases the availability of the videos in clients, which reduces the traffic demand on the server. We also observe that when the cache size is larger than 300M, the percent of server contribution in NetTube and SocialTube remains nearly constant while that in Random and PA-VOD decreases. Since the popularity of a video is about one month, each peer watches [1-4] videos per day and the video size is [20-30]M, 300M is approximately a sufficient size for a peer to keep watched videos for others' visits in NetTube and SocialTube as they can quickly identify peer servers by querying fewer nodes. Though old videos and unpopular videos are replaced by the new videos when the cache size is full, since they are unlikely to be watched, hence a larger cache size does not decrease the percent of server contribution. redIn Random and PA-VOD, a node needs to visit many peers to find a video peer server. Thus, when nodes can cache more videos, they have higher probability to find peer servers, which reduces the percent of server contribution. These results confirm the effectiveness of the buffer management algorithm in keeping videos that are most likely to be visited.

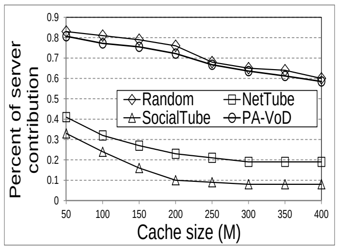


Fig. 29: Percent of server contribution vs. cache size.

8.3 The Effect of the Scope of Per-node Overlays

In this experiment, we built the per-node overlay for each source node formed by peers within x $(x = 1, 2, 3)$ social hops and tested the effect of x on video provider search efficiency and overlay maintenance cost. The

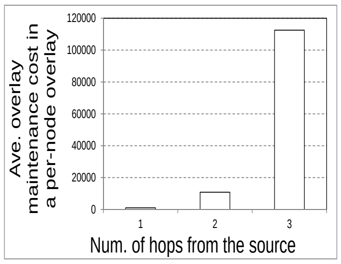


Fig. 30: Average overlay maintenance cost vs. # of social hops.

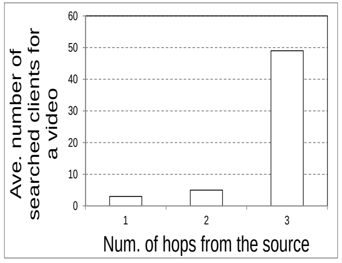


Fig. 31: Average number of searched clients for a video vs. # of social hops.

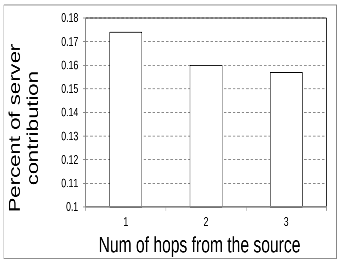


Fig. 32: Percent of server contribution vs. # of social hops.

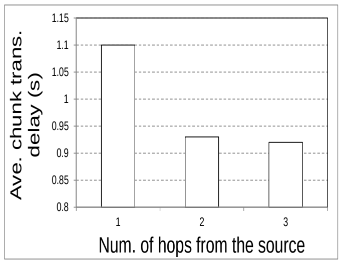


Fig. 33: Average chunk transmission delay vs. # of social hops.

overlay maintenance period was set to one simulation cycle, in which neighboring nodes exchange one message. We used a method similar as the optimistic unchoking in BitTorrent. In this method, while a node maintains 4 connections, it randomly probes a node in its swarm in every simulation cycle to search for other requested video holders that have higher uploading bandwidth, and replaces the node with the lowest uploading bandwidth in its current 4 video providers with the detected node.

Figure 30 shows the average overlay maintenance overhead of each per-node overlay of a source node when x=1, 2, 3, respectively. We see that as x increases, the average overlay maintenance overhead increases greatly, especially when x=3. As the number of hops from the source node increases (x), more nodes are included in the overlay, which generates more exchanged messages for maintenance overhead.

Figure 31 shows the average number of searched clients per video when $x = 1, 2, 3$, respectively. We see that as x increases, more clients are searched for each video. This is because when a source node's per-node overlay includes nodes within a longer social distance, more nodes need to be searched in the overlay.

Figure 32 shows the percentage of server contribution when $x = 1, 2, 3$, respectively. We see that the server contribution of $x = 3$ is not significantly lower than that of $x = 2$, which is greatly lower than that of $x = 1$. This is because most of the requested video holders are within 2 hops from the requesters as shown Figure 5. Including nodes in 3-hops from the source node does not contribute significantly in providing videos, hence the server contribution when $x = 3$ is only slightly lower than that of $x = 2$.

Figure 33 shows the average chunk transmission delay when $x = 1, 2, 3$, respectively. We see as x increases, the average chunk transmission delay decreases. The node tries to retrieve the remaining video chunks from its swarm-peers. More peers in the cluster lead to more choices of the swarm-peers, which provides more node options with high uploading bandwidth. Therefore, chunk transmission delay is reduced. We see that the transmission delay of $x = 3$ is slightly lower than that of $x = 2$. This is still because most of the nodes holding the videos are within 2 hops as shown Figure 5. Expanding the social hops to 3 hops will not generate significantly more video providers.

The above experimental results show that including the users beyond 2-hops into a source node's per-node overlay generates greater overlay maintenance cost than video sharing benefit. Therefore, in SocialTube, the per-node overlay of a source node consists of nodes within 2-hops from the source.

8.4 Overlay Maintenance Cost

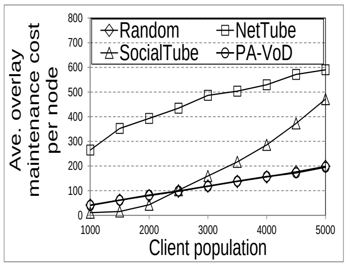


Fig. 34: Average overlay maintenance cost vs. node population.

We then evaluate the overlay maintenance of different systems. When a node finishes viewing videos in a simulation cycle, it leaves the system, and then joins in the system in the next simulation cycle. When a node joins in, it connects to previously connected nodes, and when a node leaves, it notifies its neighbors to update their neighbors. In order to prevent a node from connecting to an excessive number of nodes in each overlay/cluster, we set this upper bound to $1/50$ of the client population. Figure 34 shows the average number of messages per node for periodical probing, departures and joins in each simulation cycle. We see that NetTube generates much higher overlay maintenance cost than other systems. This is due to the fact that a node may exist in several interest clusters, so it maintains much more neighbors. SocialTube has much more overlays with much smaller overlay sizes because each video source has a per-node overlay formed by peers within 2 social hops. Due to small overlay size, the number of neighbors of each node is unlikely to reach the upper bound in each overlay; however, because SocialTube has much more overlays (approximately 3400), as the client population grows, its overlay maintenance cost increases rapidly. Random and PA-VoD have only 10 overlays with large overlay sizes, since the number of neighbors of each node in each overlay is bounded, they produce limited overlay maintenance cost. The results show that SocialTube does not generate high maintenance overhead compared to other systems.

9 Related Work

Significant research interest has been focused on P2P-based video sharing in P2P-VoD systems. The P2P-based VoD systems can be classified into unstructured P2P based VoD [3]–[18] and structured P2P based VoD [19]–[21]. In the unstructured P2P based VoD area, Wang *et al.* [3] developed a stochastic model that studied the essential behavior of P2P-VoD systems. Huang and Cheng [4], [5] studied the trace data of real-world P2P-VoD systems. Ho *et al.* [6] discussed how to apply a clustering technique to optimize the media-data replication strategies in P2P-VoD systems. Wang *et al.* [7] and Huang *et al.* [8] studied a nine-month MSN video trace and showed that peer-assistance can dramatically reduce server bandwidth costs. They also explored peer locality to reduce ISP-unfriendly traffic by restricting the P2P traffic within the ISPs. Zhou *et al.* [31] studied the heterogeneous media service in P2P-based vehicular networks. Cheng and Liu [9] studied trace data from YouTube and showed through measurements that YouTube videos form a social network with strong clustering properties based on video interest tags. They proposed to cluster all peers that have watched a particular video for video sharing among cluster peers. Li *et al.* [10] studied major features of the P2P VoD overlay networks and compared them with those in P2P file sharing and live streaming systems. Wu *et al.* [11] proposed an auction-based P2P VoD streaming strategy, which maximizes the supplying peers' bandwidth contribution by using incentive of increasing their budgets. In order to maximize utilization of peers' upload capacity, Fujimoto *et al.* [12] proposed a video-popularity-based caching and reservation scheme. Wu *et al.* [13] explored the impact of movies' popularities on the server load, and then presented both passive replacement and active push strategies to reduce the server load and achieve high QoS in VoD streaming. Zhou *et al.* [14] developed a fair sharing with bounded out-degree scheduling mechanism, which parameterizes the maximum number of peers that can serve a single request. Niu *et al.* [15] learned both human factors and system dynamics from VoD systems and used a time series techniques to predict the online population and peer upload. Chang *et al.* [16] proposed different caching strategies to achieve optimal server bandwidth consumption for both standard and high-definition channels. Wang *et al.* [17] proposed a video prefetching strategy based on user preferences to reduce video startup delay in online social networks. GoalBit [18] deploys a mesh structure network to decompose the video stream into several pieces and share between different peers.

In the structured P2P based VoD area, Yiu *et al.* [19] proposed VMesh to support interactive VoD service over the Internet. The parent nodes are formed into a DHT for chunk identification in case a peer randomly seeks a video segment to watch. Shen *et al.* [20] introduced a stable DHT ring structure embedded into a mesh-based overlay to assist chuck provider search. MBoard [21] studies the user video watching activities in a forum and proposes to form comparatively stable nodes into a stable DHT to assist multimedia sharing in forums.

In addition, a number of works utilize location information and node interest information to enhance the performance of P2P video sharing [32], [33]. Salvador *et al.* [32] proposed using peer characteristics including geographical location, peer availability and peer distance for video file sharing in BitTorrent. Gopalakrishnan *et al* [33] proposed to cluster the users with similar interests through an unsupervised clustering mechanism.

Besides VoD, P2P-based video sharing can also be found in the area of live streaming systems [34]–[36]. VUD [34] is a multi-channel live P2P video streaming system with a characteristic of decoupling peer downloading from uploading. The assignment of peer uploading is made independently of what the peer is viewing, so that the problems of channel churn and channel resource imbalance can be solved. Silva *et al.* [35] presented a hybrid strategy combining the traditional P2P and client/server architectures for the live streaming of user-generated videos. A node first requests a video from the P2P network, and if it cannot receive the video after a time period, it is redirected to the centralized server. Wu *et al.* [36] used a seven-month trace from UUSee to analyze server burden with the increasing demand imposed by multiple streaming channels. They also proposed a server capacity provisioning algorithm to optimally utilize the available server capacities between different channels with a dynamical prediction of each channel's demand.

Most research on OSNs focuses on the investigation of social network characteristics [23], [24], [28], [37]–[41]. Some research [23], [28], [37] studied and analyzed the social clustering and geographical distribution of users in Facebook. Wittie *et al.* [24] studied the user interactions on the posts in Facebook. They found that users in the same geographic region are more likely to interaction with each other. Liu *et al.* [38] studied a YouTube trace and investigated the statistic features of social networks formed by the videos. Yu *et al.* [39] studied user behaviors in P2P VoD systems, which revealed an inverse correlation between video watching time and video popularity. Puwelse *et al.* [40] presented Tribler, a social network based P2P file sharing system that exploits social phenomena as a set of extensions to BitTorrent. Kalofonos *et al.* [41] introduced MyNet, a platform for secure P2P personal and social networking services. These works leverage OSNs to improve file sharing efficiency and enable system-wide file sharing. In contrast, SocialTube leverages P2P's high efficiency to improve file sharing in OSNs and caters to the typical file sharing patterns in OSNs. To our knowledge, this is the first work to study video sharing pattern in OSNs and build a P2P-based video sharing system with a pre-fetching strategy combining friendship relations and interests in OSNs.