NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing

Xu Cheng   Jiangchuan Liu
School of Computing Science
Simon Fraser University
British Columbia, Canada
Email: {xuc, jcliu}@cs.sfu.ca

***Abstract*—The recent three years have witnessed an explosion of networked video sharing, represented by YouTube, as a new killer Internet application. Their sustainable development however is severely hindered by the intrinsic limit of their client/server architecture. A shift to the peer-to-peer paradigm has been widely suggested with success already shown in live video streaming and movie-on-demand. Unfortunately, our latest measurement demonstrates that short video clips exhibit drastically different statistics, which would simply render these existing solutions suboptimal, if not entirely inapplicable.**

Our long-term measurement over five million YouTube videos, on the other hand, reveal interesting social networks with strong clustering among the videos, thus opening new opportunities to explore. In this paper, we present NetTube, a novel peer-to-peer assisted delivering framework that explores the clustering in social networks for short video sharing. We address a series of key design issues to realize the system, including a bi-layer overlay, an efficient indexing scheme and a pre-fetching strategy leveraging social networks. We evaluate NetTube through simulations and prototype experiments, which show that it greatly reduces the server workload, improves the playback quality and scales well.

I. Introduction

The recent three years have witnessed an explosion of networked video sharing as a new killer Internet application. The most successful site, YouTube, now enjoys more than 100 million videos being watched every day [1]. An April 2008 report estimated that YouTube consumed as much bandwidth as did the entire Internet in year 2000 [2], and is still enjoying nearly 20% growth rate per month [3]. With no doubt, YouTube-like sites are changing the content distribution landscape and even the popular culture.

Further expansions of these sites however have been severely hindered by their conventional client/server architecture. Industry insiders estimate that YouTube spends roughly $1 million a day to pay for its server bandwidth [4]. This high and ever rising expense for network traffic was one of the major reasons YouTube was sold to Google [5]. It also instantly defeats any effort to lift the server bottleneck and improve user experiences. Saxena et al. reveal that the average service delay of YouTube is nearly 6.5 seconds, which is much longer than the other measured sites [6]. This enormous scalability challenge calls for a new delivering architecture beyond the basic client/server model.

This work is supported by a Canada NSERC Discovery Grant, a Strategic Project Grant, and a Nokia University Relation Initiative Grant.

In the mean time of the booming of YouTube-like sites, peer-to-peer has evolved into a promising communication paradigm for large-scale content sharing. With each peer contributing its bandwidth to serve others, a peer-to-peer overlay scales extremely well with larger user bases. Besides file sharing, it has been quite successful in supporting large-scale live streaming (e.g., CoolStreaming [7] and PPLive [8]) and on-demand video streaming (e.g., GridCast [9] and Vanderbilt VoD [10]), thus naturally being believed as an accelerator of great potentials for YouTube-like video sharing services.

Unfortunately, using peer-to-peer delivery for short video clips can be quite challenging, evidently from our latest measurement on YouTube. In particular, the length of a YouTube video is short (many are even shorter than the typical startup time in a peer-to-peer overlay), and a user often quickly loads another video when finishing the previous one, so the overlay will suffer from an extremely high churn rate. Moreover, there are a huge number of videos with highly skewed popularity, thus many of the peer-to-peer overlays will be too small to function well. They together make existing per-file based overlay design suboptimal, if not entirely inapplicable.

Our long-term measurement over five million YouTube videos, on the other hand, reveals interesting social networks among the videos. In particular, we find that the links to related videos generated by uploader's choices form a small-world network. This suggests that the videos have strong correlations with each other, which has not been explored for enhancing their streaming quality yet.

The measurement results motivate our development of *NetTube*, a novel peer-to-peer assisted streaming system customized for short video sharing. NetTube explores the social networks among the **videos** by creating a bi-layer overlay, through which peers are re-distributing the videos that they have cached. We address a series of key design issues to realize NetTube, in particular, an efficient indexing scheme that facilitates clients to efficiently locate their next video and a social network assisted pre-fetching strategy that enables delay-minimized smooth transition between video playbacks.

We have performed extensive simulations and prototype experiments to evaluate the performance of NetTube. The results show that NetTube greatly reduces the workload of server, improves the quality of playback, and scales well to large client populations.

The rest of the paper is organized as follows. Section II presents background and related work. Section III offers our latest measurement results on YouTube, motivating the development of NetTube. The architecture of NetTube is overviewed in Section IV, together with its design details. We evaluate its performance through both simulations and experiments, and the results are presented in Section V and Section VI. Finally, Section VII concludes the paper.

II. Background and Related Work

A. Online Short Video Sharing

Online videos existed long before YouTube-like sites entered the scene. YouTube and its competitors however offer integrated Web 2.0 [11] platforms for worldwide user to upload, manage, and share videos. Beyond the richer contents, the establishment of social networks is an equally or even more important factor toward their success. The videos distributed by traditional streaming media servers or peer-to-peer file downloads like BitTorrent are standalone units of content. The social network existing in YouTube however enables hyperlinks between videos, as well as communities and groups. The videos are no longer independent from each other with the clients browsing them following the links.

There has been significant research effort into understanding the workloads of traditional media servers [12], [13], [14]. While sharing similar features, we have found that many of the video statistics of YouTube are quite different from these traditional media servers, especially the video length. More importantly, the social networking aspect does not exist in these traditional media services. There are simultaneous works investigating social networks in popular Web 2.0 sites [15], [16], [17], [18]. While YouTube is also one of the targeted sites in their studies, exploring the social network for accelerating content distribution has yet to be addressed.

B. Peer-to-Peer Video Streaming

Numerous peer-to-peer protocols have been developed for live or on-demand video streaming, which can be broadly classified into two categories according to their overlay structures [19], namely, *tree-based* (e.g., ChunkySpread [20]) and *mesh-based* (e.g., PRIME [21] and Ration [22]). Both tree/multi-tree and mesh solutions have seen their success in practical deployment, and there have also been approaches that reconcile them to form hybrid overlays (e.g., LBTree [23]).

While being greatly motivated by them, we notice that these protocols share two common features: 1) each video is served by a separate overlay, with little interaction among overlays, and 2) the videos are assumed to be relatively long, typically of 1-2 hours (for movies) or even longer (for continuous TV broadcast). Though the delay for each newly joined client or a client switching to different channels has been considered as an important problem, its impact is largely confined to individual clients, and only to its initial joining phase. The YouTube-like short videos however have quite different statistics, which thus call for new solutions.

III. Measuring and Understanding Internet Short Video Sharing

In this section, we present our latest measurement results on YouTube, which facilitate the understanding of the distinct characteristics of short video sharing and hence the challenges and opportunities therein.

From March 27th to July 27th, 2008, we have crawled YouTube and obtained totaling $5,043,082$ distinct videos. This constitutes a significant portion of the entire YouTube video repository. Also, because most of these videos can be accessed from the YouTube homepage in less than 10 clicks, they are generally active and thus representative for measuring characteristics of the repository. Finally, the long period of measurement enables us to understand the dynamics of such a vigorous new site. The detailed description of measurement methodology, as well as the datasets, can be found at `http://netsg.cs.sfu.ca/youtubedata.html`.

A. Video Popularity: Scalability Challenge

We first focus on the distribution of video popularity, which has played key roles in traditional media server dimensioning, and is also critical to answer the question whether peer-to-peer is necessary in the new context.

Figure 1 shows the number of views against the rank of the video (in terms of its popularity) in log-log scale. The plot, beginning with a fit like Zipf's distribution, clearly shows that there are popular videos of much more views than others. The tail (after the 10^4 video) however decreases tremendously, indicating there are not so many unpopular videos as Zipf's predicts, likely because the links among videos enable each of them can be browsed by interested viewers through various paths, as will be illustrated later.

Figure 2 further plots the number of new videos added every week in our entire crawled dataset. This reveals the growth trend of the video uploading, which is unique to YouTube-like sites that enjoy rapid expansion. It can be seen that the growth trend can be modeled by a power law. Hence, the sheer size of the YouTube repository and its exponential growth rate would soon make the operation of such sites unaffordable, along with declining service quality to their clients. It is thus necessary to establish a new delivering architecture to reduce the server cost and to improve service quality, of which peer-to-peer is obviously a candidate of great potentials.

It is however worth noting that the measurement results do not endorse a complete removal of the server from the system. We believe that dedicated servers will still play an important role, particularly for the majority of the videos that are less frequently accessed.

B. Length of Short Videos: Stability Challenge

Our next focus is the length of YouTube videos. Whereas most traditional servers contain a significant portion of long videos, typically 1-2 hour movies (e.g., HPLabs Media Server [12]), we found that YouTube are generally much shorter, most of which (99.6%) is less than 700 seconds.

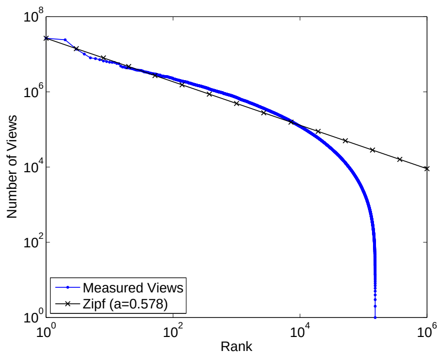

Fig. 1: Number of views against rank (in popularity) of YouTube videos

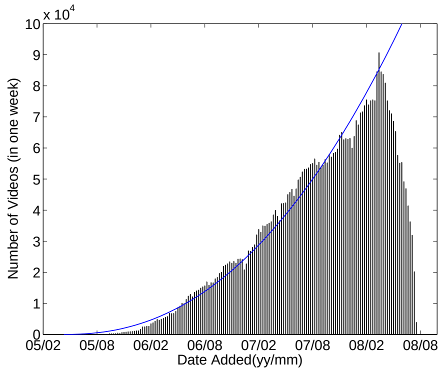

Fig. 2: Number of YouTube videos added over time for the measured datasets

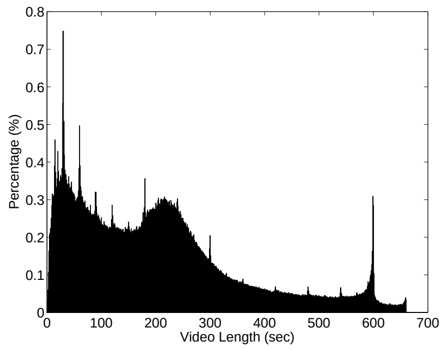

Fig. 3: Distribution of YouTube video length

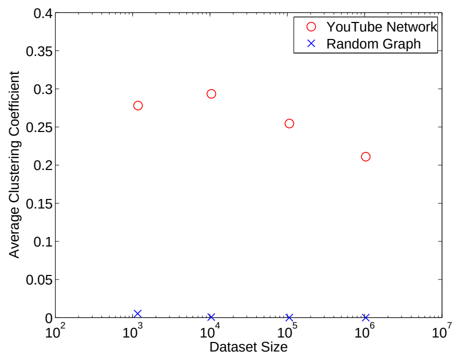

Fig. 4: Clustering coefficient of YouTube videos

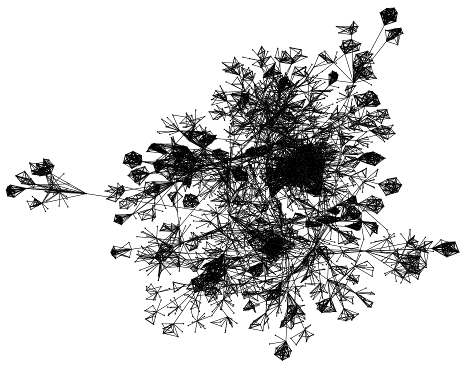

Fig. 5: A sample graph of YouTube videos and their links (directions are omitted)

Figure 3 shows the distribution of YouTube videos' lengths within 700 seconds, which exhibits three peaks. The first peak is within one minute, and contains 20% of the videos, which shows that YouTube is primarily a site for very short videos. The second peak is between 3 and 4 minutes, and contains about 17.4% of the videos. This peak is mainly caused by the large number of videos in the "Music" category, which is the most popular category for YouTube, and the typical length of a music video is often within this range [18]. The third peak is near the maximum of 10 minutes, and is caused by the limit on the length of uploaded videos. This encourages some users to circumvent the length restriction by dividing long videos into several parts, each being near the limit of 10 minutes.

Given these remarkably shorter lengths, the system is subject to much more frequent churns. Even worse, the short length is indeed close to the time scale of the initialization delays for the client joining operations in existing peer-to-peer overlays. This is particularly true for mesh-based overlays where a client has to locate and establish connections with multiple partners (often takes half minute or more). Hence, if we directly apply a conventional per-video overlays design to YouTube videos, neither the overlay nor individual clients will reach a steady state with smooth streaming quality.

C. *Social Networks of YouTube Videos: Clustering*

Finally, we examine the most distinct feature of YouTube: there are links between related videos, forming a social network among the videos. There are also communities and groups in YouTube, as well as statistics and awards for videos and personal channels, which further boost the networking. Consequently, the videos are no longer independent from each other, and the users often browse video following the network.

We have measured the network topologies formed by the related links in YouTube video pages. One of the most notable characteristics we have found is that there are strong clustering behaviors. Given the network as a graph $G = (V, E)$, the clustering coefficient C_i of a node $i \in V$ is the proportion of all the possible edges between neighbors of the node that actually exist in the graph [24]. Figure 4 shows the average clustering coefficient for the entire graph, as a function of the size of the dataset (note that the x-axis is log scale). The clustering coefficient is generally between 0.2 and 0.3, which is quite high as compared with that of the random graphs (nearly 0). A visual illustration for part of the network (about $3,000$ nodes) is shown in Fig 5, which clearly demonstrates the existence of clusters. We have also found that the average diameter (about 10) is only slightly larger than that of a corresponding random graph (about 8), which, together with the high clustering coefficients, suggests the network formed by YouTube's related videos list is a small world.

We believe that this is mainly due to the user-generated nature of the tags, title and description of the videos that is used by YouTube to form related videos. It is also worth

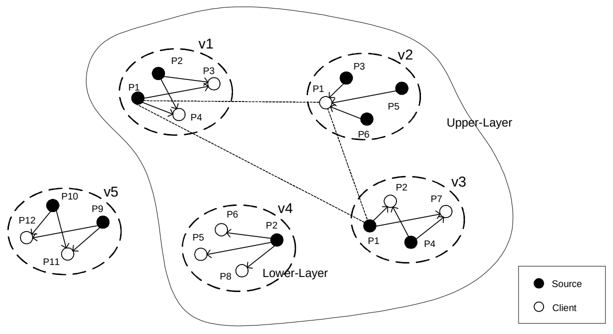

Fig. 6: Illustration of a bi-layer overlay

noting that the path length is much shorter than that in the web (18.59) [25], suggesting that the YouTube network of videos is a much closer group that is relatively easier to explore.

IV. NetTube: Architecture and Design

The above YouTube measurement motivates our development of NetTube, a novel peer-to-peer assisted delivering architecture customized for Internet short video sharing.

In NetTube, the server stores all the videos and supplies them to the clients. The clients however also share the videos through peer-to-peer communications, thus each client is referred to as a *peer*. A distinct feature of NetTube is that a peer caches all its previously played videos, and makes them available for re-distributing. The caching is practically doable given the small video sizes and the limited number of videos a rational client watches, and is indeed implemented in currently web-browsers. This is also necessary to maintain reasonable overlay sizes for peer-to-peer sharing.

As such, for a client interested in a particular video, all the peers that have previously downloaded this video serve as potential suppliers, forming an overlay for this video, together with the peers that are downloading this video. We refer to every such overlay as a *swarm*, motivated by the similar structure in BitTorrent. Obviously, a NetTube peer may appear in multiple swarms, and the server is by default in every swarm, ensuring that there is always at least one supplier. Figure 6 offers an example, where there are five swarms for videos v_1 through v_5, respectively: in the v_1-swarm, peers P_1 and P_2 are suppliers, and P_3 and P_4 are clients downloading the video.

There have been numerous algorithms for overlay construction and data swarming with standalone videos, and with minor modifications, they can be applied for each separate swarm in NetTube. Note that the cached videos have all their blocks available for sharing, which greatly simplifies the scheduling algorithm that determines which block to be fetched from which supplier with real-time constraints. Hence, NetTube does not specify any particular communication protocol for each swarm, but instead, tries to provide effective tools that facilitate its construction and maintenance.

In particular, since a client in general will watch a series of videos, it is necessary to quickly locate the potential suppliers for the next video and enable a smooth transition. To this end, NetTube introduces an upper-layer overlay on top of the swarms of individual videos. In the upper-layer overlay, given a peer, neighborhood relations are established among all the swarms that contains this peer. This is a conceptual relation that will not be used for data delivery, whereas it enables quick search for video suppliers in the social network context.

As shown in Figure 6, peer P_1 exists in swarms for v_1, v_2 and v_3, so these three swarms have mutual neighboring relations. Swarm v_4 is also in this conceptual overlay, because of the common peers P_2, P_5 and P_6. However, swarm v_5 has no relation with others given there are no common peers.

When a new peer joins the NetTube system, it first sends a request for its first video to the server. The server, keeping track of the swarms, redirects the peer to the corresponding swarm by providing a list of potential suppliers and leaves the remaining joining operations to the specific swarm construction algorithm, unless this peer is the first one to watch the video. In the latter, the server will directly provide the streaming service.

When a client finishes watching a video and begins to download the next one, it will join another swarm, but remains staying in the previous swarms as a potential supplier, until it leaves NetTube system. To join the next swarm, it first checks its partners in all its appearing swarms about the availability of the next video. If no, the partners will further check with their partners in their neighboring swarms. Intuitively, this exploits the social network's Friend-to-Friend (F2F) relation among the peers [26], because these peers in different swarms likely have close interests. For example, suppose P_4 finishes watching v_1 and clicks to v_2, its current neighbors are P_1, P_2 and P_7, but none of them has cached v_2. When P_1, P_2 and P_7 look up their neighbors' information, they find that P_3, P_5 and P_6 have cached v_2, thus P_4 can then contact with P_3, P_5 and P_6. It is social network that brings those peers closer. In the worst case, the peer can always resort to the server, though our analysis and experiments have shown that this rarely happens with large client populations.

A. Video Indexing and Searching

Each NetTube peer needs to maintain a record of its cached videos, and make it available for searching. The structure of the record has to be efficient for query and highly scalable given the sheer number of short videos. To this end, we represent each record by a Bloom filter, which is a space/time-efficient data structure for indexing. A Bloom filter [27] is an array of m bits, initially set to 0; k independent hash functions $h_1, \ldots, h_k$ with range $\{1, \ldots, m\}$ are then used to represent a set S which contains n elements. Specifically, for each element $x \in S$, the $h_i(x)$-th bit of the array is set to 1. As such, to check whether an element y is in S, we just need to check if all the $h_i(y)$-th are 1. If not, then y is clearly not in S. Otherwise, y is in S, but with false positive $(1 - \mathrm{e}^{-\frac{kn}{m}})^k$. In the NetTube context, S is the identifiers of all the cached videos at the peer. For the videoset used in our simulation, we can keep the false positive rate to be 0.0014 for $m = 256$, $k = 3$ and $n = 10$, if the hash function is perfectly random.

The server keeps track of peers' indices in an indexing table for each new peer to locate suppliers of its first video. To make the table scalable, the server will only record some of the peers as seeds instead of all the peers, and if necessary, peers will search from these seeds to find other potential suppliers of a video. The F2F social network again makes the search quite effective, as to be validated in our performance evaluation. Even if the new peer (or an existing peer) cannot locate enough suppliers, the server can serve as an additional supplier. Since this peer will playback and cache the video, the server can record it as a seed for this video.

Assume that the server keeps up to s seeds for a video. The indexing table size is then bounded by the total number of videos that have been cached in the whole system. For ease of derivation, we assume that video accesses follow the standard Zipf's distribution, and each peer on average has cached $\bar{w}$. It follows that the k-th video (ranked according to its popularity) has been accessed $v_k = \frac{1/k^a}{\mathrm{H}_{|V|,a}} \cdot \bar{w}|P|$ times, where $\mathrm{H}_{n,m} = \sum_{k=1}^{n}\left(\frac{1}{k^m}\right)$, $|V|$ is the number of all videos, and $|P|$ is the peer population. Given condition $v_k \geq 1$, we have that the total number of videos cached in the system, i.e., accessed at least once, is $\sqrt[a]{\frac{\bar{w}\cdot|P|}{\mathrm{H}_{|V|,a}}}$.

B. Social Network Assisted Pre-fetching

To achieve fast and smooth transition, we further introduce social network assisted pre-fetching in NetTube. Our experiment of over one hundred YouTube video playbacks shows that, on average, the full video can be downloaded when the playback position is 59% of the video length. That said, consider the average length is 200 second, if we pre-fetch 10 second of the video (referred to as *video prefix*), then it is sufficient to fetch 4 prefixes during the remaining playback time of the current video, given that starting to download takes half of the time. The challenge here is obviously which videos are to be fetched, so that the next video can be successfully hit. While we are facing a significant larger repository of short videos as compared to traditional repository of movies or TV programs, the next video is most likely confined by the related list of the current video. This list in general includes 20 videos at most. Therefore, even if the number of videos fetched is quite limited, the hit ratio of the next video could still be quite high.

Assume a client selects the next video in the related video list, which includes r videos, ranked from 1 through r, and the selection probability of each video is inversely proportional to its rank, that is, $\frac{1}{i\cdot s}$, where $s = \sum_{i=1}^{r}(\frac{1}{i})$. Consider the top p videos are to be pre-fetched, the probability of successful pre-fetching of the next video is $\sum_{i=1}^{p}(\frac{1}{i\cdot s})$. Figure 7 plots the estimated pre-fetching accuracy as a function of number of pre-fetches for different numbers of related videos. For the average number of related videos being 10, pre-fetching 3 video prefixes can achieve an accuracy of 62%, with a cost of about 1 MB for typical YouTube videos.

While this pre-fetching accuracy is not exceptionally high, we emphasize that it is for the pre-fetching during one

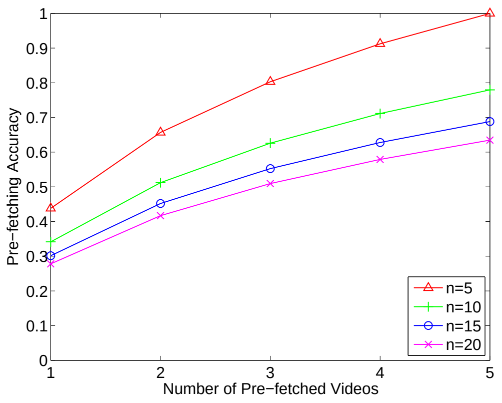

Fig. 7: Pre-fetching accuracy as a function of the number of pre-fetches

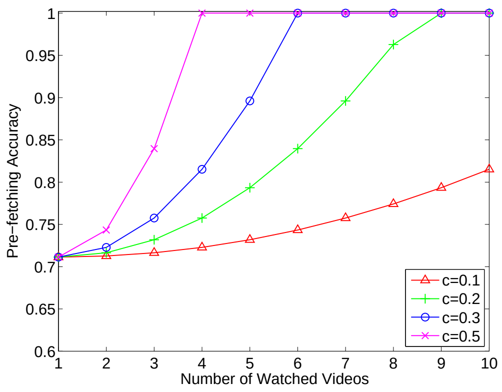

Fig. 8: Pre-fetching accuracy as a function of the number of watched videos

video playback. With the existence of social networks, the effectiveness of pre-fetching can be much higher after a client playbacks multiple videos. The intuition is that, given that the videos are generally clustered, the videos pre-fetched for the current video are likely in the related list of the next video, and so forth. Hence, as long as the client keeps all the pre-fetched prefixes, which consumes a small disk space, the chance that a prefix to be pre-fetched has already in its cache will only increase over the time. That is, the client can skip it and instead fetch another video's prefix, and the pre-fetching accuracy will accordingly increase.

To illustrate this, we now offer a simple analysis. We assume that each time a peer pre-fetches p distinct video prefixes, so that when the peer starts pre-fetching after finishes downloading the i-th videos, it has stored $((i-1)\cdot p)$ video prefixes. Let c be the ratio that these videos are in the related list of the current video, which is a factor depending on the clustering coefficient.[1] Since the newly pre-fetched p video prefixes are among the top list, the accuracy can be evaluated by $\sum_{i=1}^{p}(\frac{1}{i\cdot s}) + \frac{\min(r-p,(n-1)pc)}{r-p}\sum_{i=p+1}^{r}(\frac{1}{i\cdot s})$, even if the intersection $((i-1)\cdot p\cdot c)$ video prefixes are only uniformly

[1]Our simulations show that, for the crawled datasets, the average of c is 0.3225 with a median of 0.3061.

distributed in the rest of the list.

Figure 8 plots the estimated pre-fetching accuracy as a function of number of watched videos for different c, where p is set to 4. Clearly, when the number of watched videos increases, the pre-fetching accuracy increases, so for larger c. Given the typical c of 0.3, the accuracy becomes 0.9 after playback 5 videos.

V. Performance Evaluation

We have evaluated the performance of NetTube through both simulations and prototype experiments. We first present the simulation results in this section.

A. *Configuration*

Our simulation is based on part of our latest crawled dataset (on April 2nd, 2008), which contains $6,951$ distinct videos, each having at least one and up to 20 related videos, which is a representative subset of the data that makes the simulation time-efficient, and we have confirmed that its properties closely resemble that of the whole dataset. To emulate the real Internet environment, the peers are heterogenous in terms of bandwidth, roughly following the statistics in [14] as shown in Table I.

	I	II	III
Downloading	768 kbps	1536 kbps	3072 kbps
Uploading	128 kbps	384 kbps	768 kbps
Distribution	21.4%	23.3%	55.3%

TABLE I: Bandwidth capacity and distribution of users

The video bitrate is set to 330 kbps, typical for YouTube videos [18], and to support a smooth streaming, the minimum streaming rate is 384 kbps, which is recommended by [28]. Based on empirical observations, we assume that, whenever buffer underflow occurs, the peer will pause for 3 seconds and then resume playback.

We assume that a client randomly selects the first video within a list of popular and pre-defined videos. Each next video is then selected from the related video list with a probability that is inversely proportional to the order of that video, and a video that has already been watched will be skipped. We also assume that there is a 5 seconds gap between finishing watching the current video and selecting the next, and the number of videos watched by a peer follows a normal distribution with an average of 10 videos. The above model is largely based on empirical observations. Even though there is no publicly-available log from the YouTube administrator so far, we believe that our model well captures the essences of YouTube clients' behaviors.

B. *Server Bandwidth Reduction*

The most important objective of NetTube is to reduce the server load. To understand the effectiveness of NetTube in this respect, we have also implemented the client/server system and Peer-Assisted VoD (PA-VoD) [14], a state-of-the-art system that relieves server stress of MSN videos. In PA-VoD, the clients also serve as peers to relay traffic, but unlike in NetTube, they do not cache and share videos that they have downloaded. It is also blind to the existence of social networks. More detailed description of PA-VoD can be found in [14].

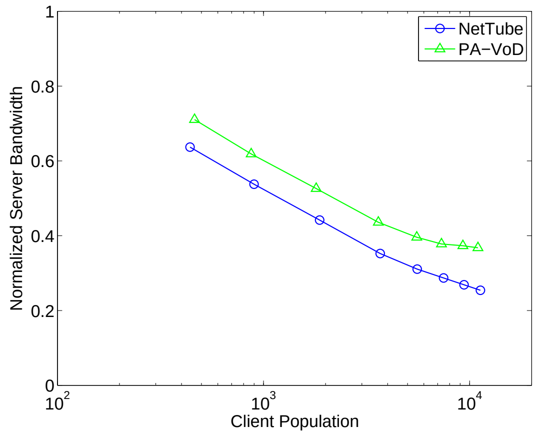

Fig. 9: Server bandwidth consumptions of NetTube and PA-VoD (normalized by the total bandwidth in the client/server model)

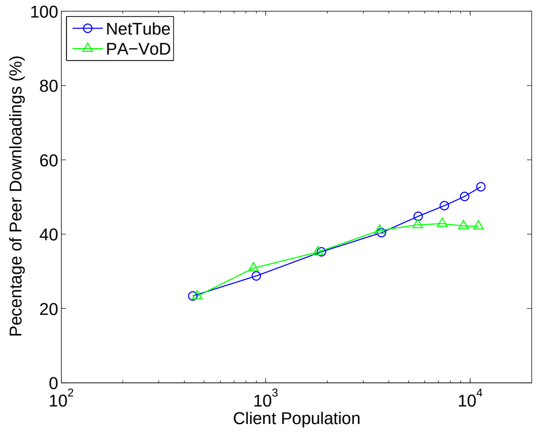

Fig. 10: Percentage of peer downloadings (as opposite to downloadings from the server) in NetTube and PA-VoD

Figure 9 compares the server bandwidth consumptions of NetTube and PA-VoD, where both are normalized by the total bandwidth of the pure client/server system. It is obvious that NetTube saves significantly more server bandwidth than PA-VoD does for all client populations. More importantly, when the number of clients increases, the consumption of NetTube drops more quickly than that of PA-VoD. For 12000 online clients, it requires only 25% of the bandwidth as with client/server, suggesting that NetTube is quite scalable with client population.

To further understand the savings, we examine the distribution of the downloading sources in the overlays. Figure 10 plots the percentage of peers that are downloading only from other peers, while not from the server. We can see that when the number of clients is less than four thousand, NetTube and PA-VoD behave similarly. However, when the online clients increase, more and more NetTube peers do not download

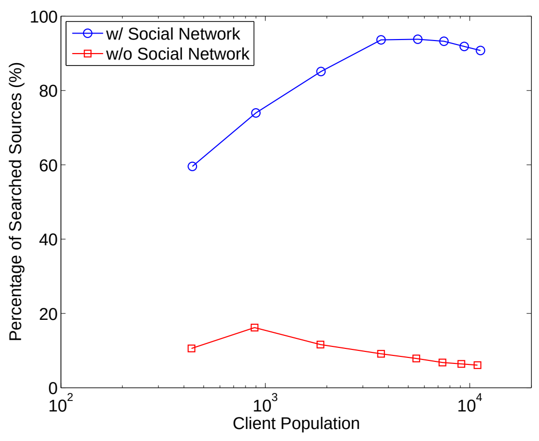

Fig. 11: Percentage of searched sources

from server but from other peers. This is however not the case in PA-VoD, where the percentage is relatively constant. It again confirms that the better scalability of NetTube and the effectiveness of NetTube's cache design.

C. *Impact of Social Networks*

We next examine how NetTube benefit from the existence of the social networks. To this end, we create a dataset with independent videos yet preserving the popularity distribution as the YouTube dataset used in the previous experiments. The next video is thus not confined by the related lists, but is selected from the whole repository following the same popularity distribution. We then ran NetTube on this setting and compared the results with those with social networks.

For each to playback by a client, we record the number of source peers returned by the search among the client's neighbors (see Section IV). We then calculate the percentage of sources returned from neighbors over the total number of the available sources, including all other clients that have cached the video. Figure 11 compares the percentages with and without social network.

We can see in the simulation with social networks, the percentage increases fast before the online peers reaches 3000, and reaches nearly 95%. It slowly drops as population keeps growing. This is because once a client has found enough sources for the expected video, it can simply cancel the search, resulting in lower percentage with large client population. This in turn suggests that the existence of social networks indeed makes our F2F-based search quite effective, despite its simplicity and limited search range (2 hops only). In comparison, if no social network exists, peers can only find less than 20% sources, and this percentage keeps decreasing.

We further plot in Figure 12 the probability that a peer's neighbors have the video to be pre-fetched. We can see that the probability in the simulation with social network (0.7 when online peers are more than 7000) is much higher than that without social network (nearly 0), confirming that with social network, neighbors are more likely to share similar interests. With more neighbor peers having the predicted video, it is the

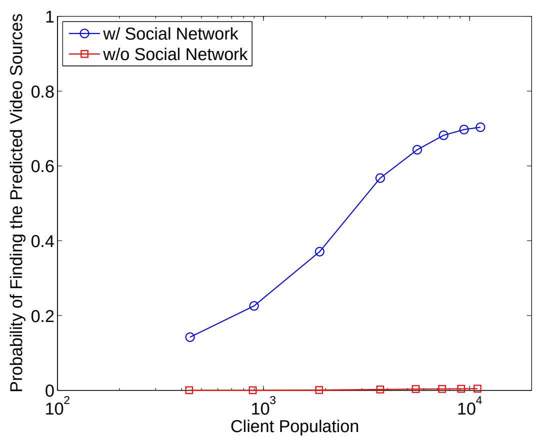

Fig. 12: Probability of neighbor having the predicted video

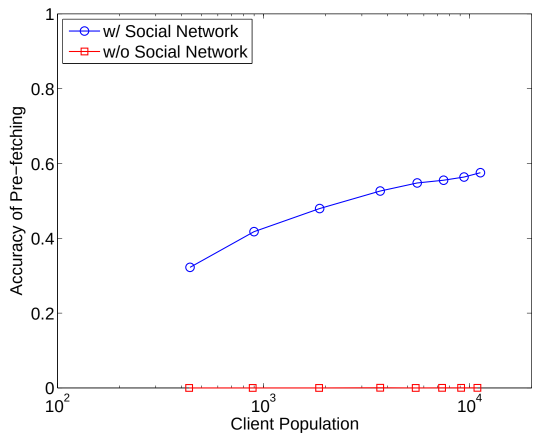

Fig. 13: Accuracy of pre-fetching

basis of our pre-fetching algorithm.

Figure 13 compares the accuracy of pre-fetching in the simulations with and without social network. The accuracy in the simulation with social network is about 55%, which is much higher than that in the simulation without social network (nearly 0). The latter is so low simply because the video repository is too large and a small number of tries (say 10) can hardly hit the next video if the videos are independent. In other words, given the huge number of the whole repository of YouTube videos, which is orders of magnitude higher than that for the simulation, it is close to impossible to design an effective pre-fetching strategy if we are blind to social networks. We note that the accuracy of NetTube is lower than our analysis in Section IV-B, mainly because when peer cannot find a supplier for a predicted video, it has to try a lower ranked video in the list. However, we also find that as the number of online peers increases, the accuracy increases, because with larger user base, more videos are likely to be downloaded and cached.

D. *Effectiveness of Indexing*

Finally, we study the effectiveness of our indexing scheme. We first focus on the false positives, which increases the

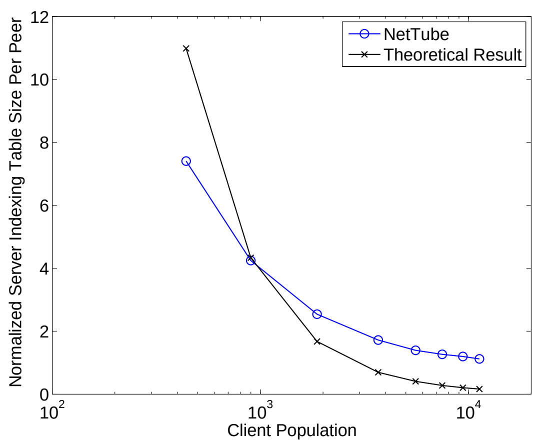

Fig. 14: Normalized server indexing table size per peer

overhead and hence the startup delay. We find that the false positive rate in the simulation is around 0.02. This is higher than the theoretical result of 0.0014 with the given parameters, mainly because the hash function is not perfectly random. Nevertheless, it is still acceptable, and more importantly, is independent of the client population.

Next, we examine the indexing table size at the server. As mentioned earlier, besides the number of recorded seeds, the number of videos that have been watched also affects the size of server's indexing table size. Therefore, we record the percentage of watched videos against all the videos, and normalized the indexing table size per peer by this percentage as *normalized size*. We plot the result in Figure 14, together with the theoretical result. We can see the normalized indexing table size quickly decreases with increasing online peers, suggesting indexing design is quite scalable. The theoretical results also roughly fit the simulation results reasonably well. We believe the difference is mainly because the Zipf distribution used in analysis does not perfectly reflect the popularity distribution.

VI. Prototype Experiments over PlanetLab

To further investigate the performance of NetTube, we have implemented a prototype and conducted experiments over the PlanetLab network [29].

The experiment is again based on our latest crawled dataset. Given the relatively small scale of the current PlanetLab, we use a smaller set of 500 distinct videos, which is a subset of the data we used for simulation. We install the server at `199.60.13.45`. To emulate larger client bases, we let the client program on each PlanetLab node run twice at different times. We have conducted a series experiment with the number of online nodes ranging from 50 to 235. Other configurations are similar to that in the simulations, except that if a client suffers severe playback delay (60 second), it will fail and the client program will restart. The failure is probably due to the insufficient networking and system resource of PlanetLab nodes.

We again look at the bandwidth consumption of the server first. To this end, we record the total downloading traffic and

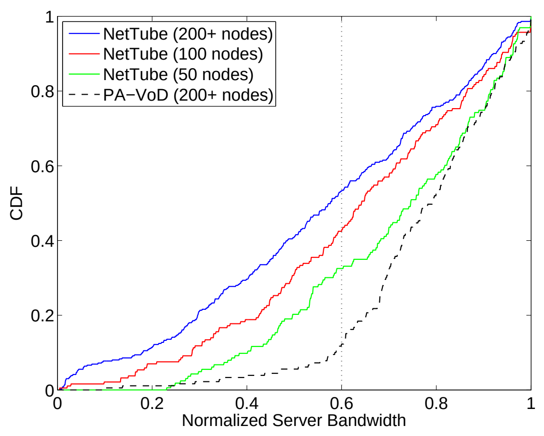

Fig. 15: Cumulative distribution of normalized server traffic

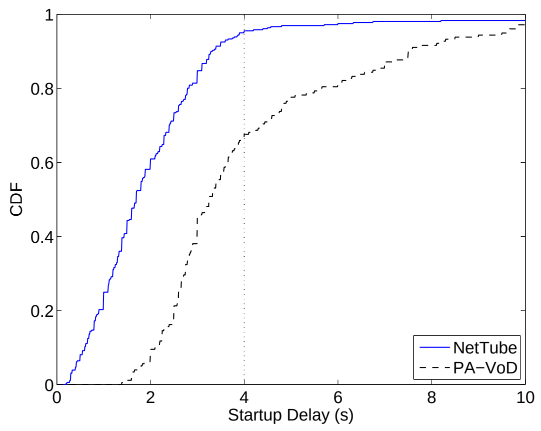

Fig. 16: Cumulative distribution of startup delays

total downloading traffic from server for each client. We plot the cumulative distribution function (CDF) of the normalized server bandwidth in Figure 15. Clearly, NetTube saves more server bandwidth: for 200 concurrent clients, near 55% of them have downloaded less than three fifths traffic from server, and the numbers are 45% and 33% peers with 100 and 50 clients, respectively. We can naturally expect that the savings will be more significant with larger client populations, as validated by our simulation results. In contrast, only 10% peers save that much bandwidth with PA-VoD.

We next examine the startup delay, that is, the interval from selecting a new video to starting play this video. We plot the CDF of average startup delay in Figure 16. Since in PA-VoD, the peers request peer list only from server, while in NetTube, peers request peer list only from neighbors except for the first video, it should have taken more time for NetTube to start watching a video. However, NetTube behaves much better than PA-VoD: more than 95% peers have an average startup delay shorter than 4 seconds in NetTube, whereas only 70% peers can achieve this in PA-VoD. The average delay for NetTube is 2.18 second, which is much lower than that of PA-VoD, 4.41 second. We believe that this is mainly due to the high success ratio of the social network assisted pre-fetching.

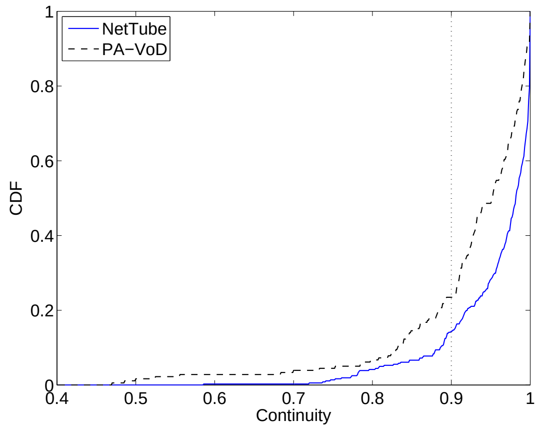

Fig. 17: Cumulative distribution of playback continuity

Finally, we investigate the playback quality experienced by individual clients, in particular, the continuity of the streaminglized playback as represented by the fraction of the on-time data blocks [7]. In Figure 17, we plot the CDF of playback continuity, which shows that, in NetTube, near 90% of the clients enjoy a continuity over 0.9, and 70% even achieve 0.95 continuity. In contrast, only 50% of the PA-VoD client can achieve such a level, and there are over 2% peers suffering from continuity less than 0.5. This is mainly because those peers have to wait the same time as the playback time, thus having much longer delays. Such results confirm that NetTube better accommodate churns introduced by frequent transitions among short videos.

VII. Conclusion

YouTube-like sites have become one of the killer Internet applications recently, and peer-to-peer delivering has been considered as an alternative to their deficient client/server architecture. In this paper, we however presented strong evidence that this new generation of short video sharing service has quite unique statistics, and the existing peer-to-peer streaming systems designed for traditional video contents thus will not work well in this new context. On the other hand, we also showed through measurements that the YouTube videos form a social network with strong clustering properties, which does not exist in traditional video sites either.

We then proposed NetTube, a novel peer-to-peer assisted delivering framework leveraging the social network, in particular, clustering, for short video sharing. We addressed a series of key design issues to realize the system, including a bi-layer overlay, an efficient indexing scheme and a prefetching strategy benefiting from social networks. Its performance was evaluated through both simulations and PlanetLab experiments, which demonstrates the necessity and superiority of utilizing social networks.

There are many possible future venues toward enhancing NetTube. We are particularly interested in developing more efficient and scalable indexing schemes to speed up searching of video suppliers. We are also interested in further understanding the characteristics of the YouTube social networks as well as client behaviors for improving our prototype implementation.

References

[1] "YouTube serves up 100 million videos a day online," http://www.usatoday.com/tech/news/2006-07-16-youtube-views_x.htm, 2006.

[2] "Web could collapse as video demand soars," http://www.telegraph.co.uk/news/uknews/1584230/Web-could-collapse-as-video-demand-soars.html, 2008.

[3] C. Corbett, "Peering of Video," http://www.nanog.org/mtg-0606/pdf/bill.norton.3.pdf, 2006.

[4] "YouTube looks for the money clip," http://techland.blogs.fortune.cnn.com/2008/03/25/youtube-looks-for-the-money-clip/, 2008.

[5] "Google to buy YouTube for $1.65 billion," http://money.cnn.com/2006/10/09/technology/googleyoutube_deal/index.htm, 2006.

[6] M. Saxena, U. Sharan, and S. Fahmy, "Analyzing Video Services in Web 2.0: A Global Perspective," in *Proc. of ACM NOSSDAV*, 2008.

[7] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," in *Proc. of IEEE INFOCOM*, 2005.

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

[9] B. Cheng, L. Stein, H. Jin, and Z. Zhang, "A Framework for Lazy Replication in P2P VoD," in *Proc. of ACM NOSSDAV*, 2008.

[10] B. Chang, L. Dai, Y. Cui, and Y. Xue, "On Feasibility of P2P On-Demand Streaming via Empirical VoD User Behavior Analysis," in *Proc. of IEEE ICDCS*, 2008.

[11] T. O'Reilly, "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software," http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2005.

[12] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "Long-term Streaming Media Server Workload Analysis and Modeling," HP Labs, Tech. Rep., 2003.

[13] E. Veloso, V. Almeida, J. Wagner Meira, A. Bestavros, and S. Jin, "A Hierarchical Characterization of a Live Streaming Media Workload," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 1, pp. 133–146, 2006.

[14] C. Huang, J. Li, and K. Ross, "Can Internet Video-on-Demand be Profitable?" in *Proc. of ACM SIGCOMM*, 2007.

[15] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *Proc. of ACM IMC*, 2007.

[16] A. Mislove, M. Marcon, K. Gummadi, P. Dreschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proc. of ACM IMC*, 2007.

[17] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View From the Edge," in *Proc. of ACM IMC*, 2007.

[18] X. Cheng, C. Dale, and J. Liu, "Statistics and Social Network of YouTube Videos," in *Proc. of IEEE IWQoS*, 2008.

[19] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, 2008.

[20] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast," in *Proc. of IEEE ICNP*, 2006.

[21] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming," in *Proc. of IEEE INFOCOM*, 2007.

[22] C. Wu, B. Li, and S. Zhao, "Multi-Channel Live P2P Streaming: Refocusing on Servers," in *Proc. of IEEE INFOCOM*, 2008.

[23] F. Wang, J. Liu, and Y. Xiong, "Stable Peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming," in *Proc. of IEEE INFOCOM*, 2008.

[24] D. Watts, *Small Worlds: the Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[25] R. Albert, H. Jeong, and A.-L. Barabási, "The Diameter of the World Wide Web," *Nature*, 1999.

[26] D. Bricklin, "Friend-to-Friend Networks," http://www.bricklin.com/f2f.htm, 2000.

[27] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[28] "Creating Flash Video (FLV) Files with Flash Video Exporter," http://www.adobe.com/devnet/flash/articles/flv_exporter_02.html, 2004.

[29] "PlanetLab," http://www.planet-lab.org.