Exploring Interest Correlation for Peer-to-Peer Socialized Video Sharing

XU CHENG and JIANGCHUAN LIU, Simon Fraser University

The last five 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, reveals interesting social networks with strong correlation among the videos, thus opening new opportunities to explore. In this article, we present NetTube, a novel peer-to-peer assisted delivering framework that explores the user interest correlation 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, a delay-aware scheduling mechanism, and a prefetching strategy leveraging interest correlation. We evaluate NetTube through both simulations and prototype experiments, which show that it greatly reduces the server workload, improves the playback quality and scales well.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Design; Measurement; Performance

Additional Key Words and Phrases: YouTube, video on demand, peer-to-peer, social network

ACM Reference Format:

Cheng, X. and Liu, J. 2012. Exploring interest correlation for peer-to-peer socialized video sharing. ACM Trans. Multimedia Comput. Commun. Appl. 8, 1, Article 5 (January 2012), 20 pages.
DOI = 10.1145/2071396.2071401 http://doi.acm.org/10.1145/2071396.2071401

5

1. INTRODUCTION

The last five years have witnessed an explosion of networked video sharing as a new killer Internet application. The most successful site, YouTube, now enjoys more than 20 hours of videos being uploaded every minute by worldwide users [Blog 2009] and reaches all-time high of 14.6 billion videos viewed in May, 2010 [Rao 2010]. An April 2008 report estimated that YouTube consumed as much bandwidth as

This research was supported by a Canada NSERC Strategic Project Grant, an NSERC Discovery Grant, an NSERC DAS Grant, and an MITACS Project Grant.
Authors' address: X. Cheng and J. Liu, School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada; email: jeliu@cs.sfu.ca.

DOI 10.1145/2071396.2071401 http://doi.acm.org/10.1145/2071396.2071401

did the entire Internet in year 2000 [Carter 2008], and is enjoying nearly 20% growth rate per month [Corbett 2006]. 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 estimated that YouTube spent roughly $1 million a day to pay for its server bandwidth [Yen 2008]. This high and ever rising expense for network traffic was one of the major reasons that YouTube was sold to Google [La Monica 2006]. It also instantly defeats any effort to lift the server bottleneck and improve user experiences. Saxena et al. [2008] revealed that the average delay of YouTube was much longer than the other measured sites, and `Alexa.com` [Alexa 2010] comfirmed 58% of other sites were faster. This enormous scalability challenge calls for a new delivering architecture beyond the basic client/server model.

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 [Zhang et al. 2005] and PPLive[1]) and on-demand video streaming (e.g., GridCast[2]), 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 large-scale 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 a peer-to-peer overlay will suffer from an extremely high churn rate. Given the shorter length, the startup and playback delay would be effectively amplified from the perception of users, too. Moreover, there are a huge number of videos with highly skewed popularity, and 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 interest correlation by creating a bi-layer overlay, through which peers re-distribute 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, a delay-aware scheduling mechanism to improve the playback quality, and a cluster-aware prefetching 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 article is organized as follows. Section 2 presents the background and related work. Section 3 offers our latest measurement results on YouTube, motivating the development of NetTube. The architecture of NetTube is overviewed in Section 4, together with its design details in Section 5.

[1]http://www.pplive.com.
[2]http://www.gridcast.cn.

We evaluate its performance through both simulations and experiments, and the results are presented in Section 6 and Section 7, respectively. Finally, Section 8 concludes the article.

2. BACKGROUND AND RELATED WORK

2.1 Online Video Sharing and Web 2.0 Applications

Online videos existed long before YouTube-like sites entered the scene. YouTube and its competitors, however, offer integrated Web 2.0 [O'Reilly 2005] 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 stand-alone units of content. The social network existing in YouTube, however, enables hyperlinks between videos, as well as the external links from other sites [Lai and Wang 2010]. 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 [Huang et al. 2007; Cha et al. 2008; Yin et al. 2009]. 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, there is no correlation among videos in these traditional media services.

In recent years, there have been simultaneous works investigating social networks in popular Web 2.0 sites, such as YouTube, Facebook, MySpace, Flickr, and LinkedIn [Cha et al. 2007; Gill et al. 2007; Mislove et al. 2007; Cheng et al. 2008, 2010; Saxena et al. 2008; Benevenuto et al. 2009; Schneider et al. 2009; Song et al. 2009]. The focuses were mainly on Web application statistics, user behavior, and network usage. While YouTube has been also one of the targeted sites in their studies, exploring the social network particularly the video interest correlation for accelerating content distribution has yet to be addressed.

2.2 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 [Liu et al. 2008], namely, *tree-based* (e.g., ChunkySpread [Venkataraman et al. 2006]) and *mesh-based* (e.g., Coolstreaming [Zhang et al. 2005] and PRIME [Magharei and Rejaie 2009b]). Both tree/multitree and mesh solutions have seen their success in deployment, and there have also been many practical applications, such as PPLive P2P-VoD [Huang et al. 2008], CPM [Gopalakrishnan et al. 2009], and etc. Many studies have been conducted on different perspectives of peer-to-peer video streaming, such as multichannel streaming [Wu et al. 2008, 2009], exploring stable/superior peers [Wang et al. 2008; Liu et al. 2009], scheduling in live/VoD streaming [Aggarwal et al. 2009; Liang et al. 2009], and overlay maintenance [Magharei and Rejaie 2009a; Qiu et al. 2009].

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. We have also extended our previous work [Cheng and Liu 2009] through improving the indexing scheme and proposing a new delay-aware scheduling mechanism, and our latest experiment confirms that the new design greatly improves the original NetTube.

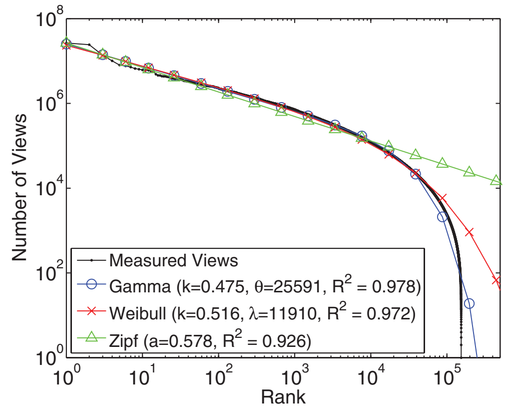

Fig. 1. Number of views against rank (in popularity) of YouTube videos.

3. MEASURING AND UNDERSTANDING INTERNET SHORT VIDEO SHARING

In this section, we present our latest large-scale 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 27 to July 27, 2008, we have been crawling YouTube and obtained $5,043,082$ distinct videos in total. This constitutes a good 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 the 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 can be referred to our previous work [Cheng et al. 2008; Cheng et al. 2010], and the datasets can be found at `http://netsg.cs.sfu.ca/youtubedata.html`.

3.1 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 numbers of views against the ranks of the videos (in terms of their popularities) in log-log scale. Note that we collected a number of datasets in different days, and since the number of views is a dynamic property, we used only one of the dataset to measure, which is relatively static. Therefore, there are around 100 thousand videos measured for this property, as shown in the figure. 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 2×10^4 videos) however decreases tremendously, indicating that there are not so many unpopular videos as Zipf's predicts, likely because the links among videos enable each of them to 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 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

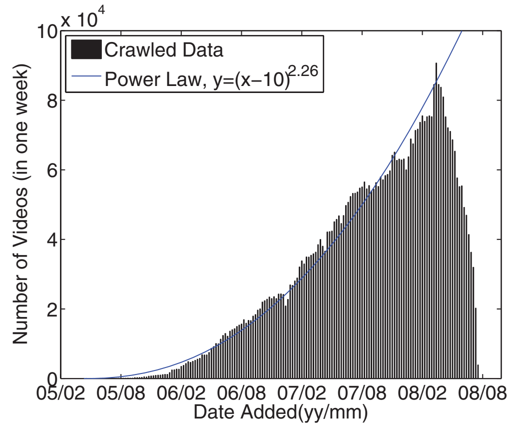

Fig. 2. Number of YouTube videos added over time for the measured datasets.

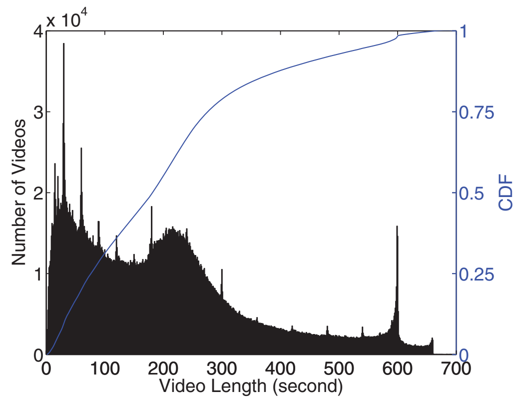

Fig. 3. Distribution of YouTube video length.

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.

3.2 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, we found that YouTube videos are generally much shorter, most of which (99.6%) are less than 700 seconds.

Figure 3 shows the distribution of YouTube videos' lengths within 700 seconds, which exhibits three peaks. The first peak is around 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 [Cheng et al. 2008, 2010]. The third peak is near the maximum of 10 minutes,

Fig. 4. A sample graph of YouTube videos and their links (directions are omitted).

and is obviously 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 clients 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). The shorter length brings more stringent delay requirements, because any delay, even being minor as compared to those for conventional movie-like videos, will be perceptually amplified. Hence, if we directly apply a conventional per-video overlays design to YouTube videos, neither the overlay nor individual clients would reach a steady state with smooth streaming quality.

3.3 Correlation Among Videos: Opportunity

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 external sites directing to YouTube videos [Lai and Wang 2010], 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. A visual illustration for part of the network (about 4,000 nodes) is shown in Figure 4. 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 [Watts 1999]. Figure 5 shows the average clustering coefficient for the entire graph, as a function of the size of the dataset (note that the x-axis is of log scale). The clustering coefficient is generally between 0.2 and 0.3, which is quite high as compared with that of random graphs (nearly 0). In Figure 6, we have also found that the average characteristic path length (about 10) is only slightly larger than that of a corresponding random graph (about 8), which, together with the high clustering coefficients, suggests that the network formed by YouTube's related videos list is a small world. This can also be observed from the visual illustration (Figure 4).

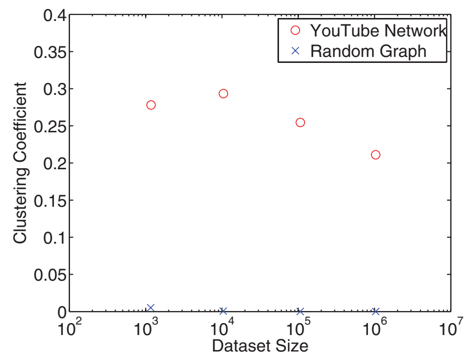

Fig. 5. Clustering coefficient of YouTube videos.

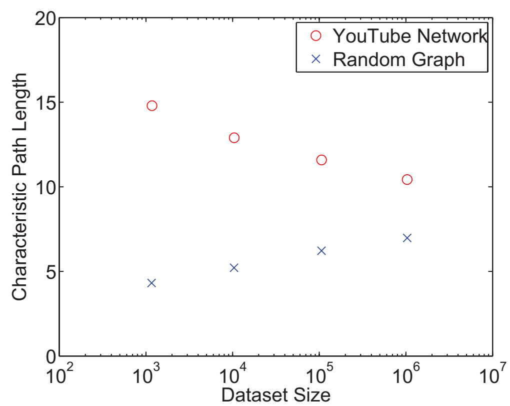

Fig. 6. Characteristic path length of YoutTube videos.

We believe that this is mainly due to the user-generated nature of the tags, title and description of the videos that are used by YouTube to relate videos. It is also worth noting that the path length is much shorter than that in the Web (18.59) [Albert et al. 1999], suggesting that the YouTube network of videos is a much closer group that is relatively easier to explore.

4. NETTUBE: OVERVIEW AND ARCHITECTURE

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

In NetTube, the server stores all the videos and supplies them to clients. The clients also share the videos through peer-to-peer communications, and 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 redistributing. 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. It 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 can 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

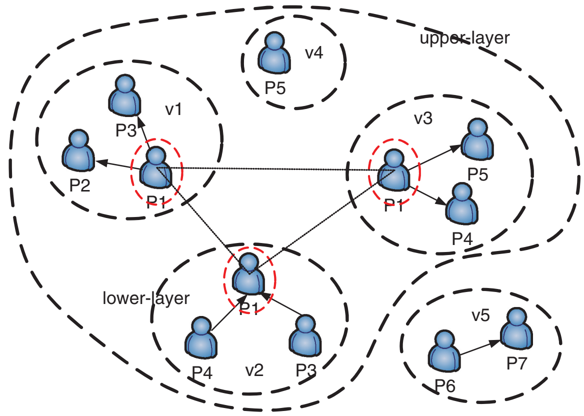

Fig. 7. Illustration of a bi-layer overlay.

is by default in every swarm, ensuring that there is always at least one supplier. Figure 7 offers an example, where there are five swarms for videos, namely v_1 through v_5. In the v_1-swarm, peers P_1 is a supplier, and P_2 and P_3 are clients downloading the video.

There have been numerous algorithms for overlay construction and data swarming with standalone videos. 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; instead it enables quick search for video suppliers in the social network context. As shown in Figure 7, peer P_1 exists in swarms for v_1, v_2 and v_3, so these three swarms have mutual neighboring relations, connected by peer P_1. Swarm v_4 is also in this conceptual overlay, because of the common peer P_5. 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 in the previous swarms as a potential supplier, until it leaves the 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 video interest correlations among the peers, because these peers in different swarms likely have close interests. For example, suppose P_2 finishes watching v_1 and clicks to v_2, it has only one current neighbor, P_1. P_1 has not cached v_2, yet when P_1 looks up its neighbors' (all the peers it has seen in the overlays) information, it finds that P_3 and P_4 have cached v_2. Thus P_1 can

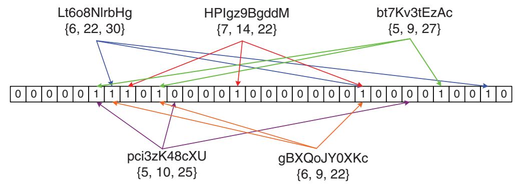

Fig. 8. Illustration of a peer's index with $m = 32, k = 3$.

ask P_2 to contact with P_3 and P_4, and P_2 can then download v_2 from them. It is interest correlation 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.

5. NETTUBE DESIGN ISSUES

5.1 Video Indexing and Searching

In the system, each NetTube peer needs to maintain a record of its cached videos, and makes it available for searching. The server also needs to keep track of peers' videos. Hence the indexing structure of the record has to be efficient for query and highly scalable given the sheer number of short videos. To this end, we propose a space-efficient Bloom filter index, and a fast-searching indexing table.

Bloom filter is a probabilistic data structure for representing a set of elements [Bloom 1970], and has been widely adopted in different applications such as distributed caching, peer-to-peer/overlay networks and resource routing. Each Bloom filter 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 bits are 1. If not, then y is clearly not in S. Otherwise, y is likely in S, but with a false positive probability of $(1 - \mathrm{e}^{-\frac{kn}{m}})^k$. In the NetTube context, S is the identifiers of all the cached videos at the peer.

We represent each record, that is, a series of video IDs, by a Bloom filter. Each time a peer finishes downloading a video, the video ID is mapped to update the Bloom filter index. Figure 8 illustrates an example of the Bloom filter index recorded by a peer. Each of the cached videos ("Lt6o8NIrbHg," "HPlgz9BgddM," and "bt7Kv3tEzAc") is mapped to three bits, which are set to 1. When the peer receives a request for any of the three cached videos, it can easily confirm that it caches the video. When the peer receives a request for video "pci3zK48cXU," its finds out that not all of the mapped bits (5, 10, and 25) are 1, and thus it can be assured that this video is not cached. However, if it receives a request for video "gBXQoJY0XKc," the peer will find that all of the mapped bits (6, 9, and 22) are 1. It will assume it caches the video but actually does not, resulting in a false positive.

For fast searching, we further design an indexing table to organize a list of peer records (Bloom filter indexes), which is shown in Figure 9. The server utilizes this table to keep track of peers' indices for each new peer to locate suppliers of its first video. To make the table scalable, the server records only some of the peers as seeds instead of all the peers, and if necessary, a peer can search from these seeds to find other potential suppliers of a video. The video interest correlation 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.

5.2 Delay-Aware Scheduling

The system adopts a mesh-based overlay organization, where the peers pull expected data from a set of partners (other peers or the server) through a sliding-window-based scheduling algorithm. Similar

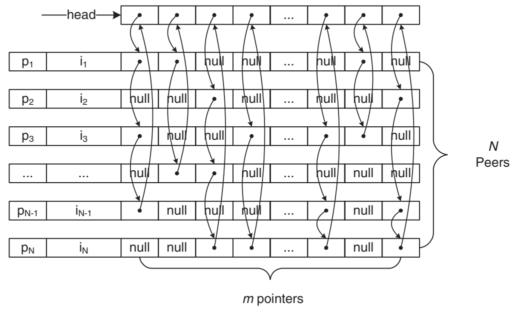

Fig. 9. Illustration of an indexing table.

algorithms have been widely used in other peer-assisted live or on-demand streaming systems [Liu et al. 2008]. Yet the short video brings new challenges to the scheduler design. First, given the shorter video length, the startup and playback delay would be amplified from the perception of users; second, given the users are less patient in waiting for short videos, more trials of joining/leaving would occur, leading to higher churn rates.

To address these challenges, we propose a novel delay-aware scheduling that is customized for the short videos. Specifically, we implement an intelligent indicator in the downloading buffer to tell whether the peer is about to encounter delay. If yes, we will utilize an aggressive strategy for transmitting the data to mitigate delay. That is, the senders will prioritize such requests, even if they have to suspend some other transmissions.

We now elaborate the detailed design of the scheduler. Suppose the size between the head of the sliding-window and the indicator is L. When the buffer within L is not full, it indicates that the peer is suffering or about to suffer from delay, because some pieces near the playback position are missing.

It is intuitive that a large L will increase the server workload. In particular, if the peer suppliers cannot provide enough streaming rate, the server has to serve the peer. The more the peer requests, the more the workload of the server will be. On the other hand, L should be large enough for the peer to detect and request the missing data in time. We now derive the lower bound of L, so as to guarantee the playback continuity. When a peer detects some pieces in the last second of the buffer within L being missing, it needs to recover the missing pieces in $(L-1)$ seconds to avoid any delay. Assume the receiver needs t seconds to detect and request to the suppliers, we have

$$(L-1-t)\cdot\left(\sum r + r_S\right) > 1 \cdot r_B.$$

Thus the minimum L is

$$L > \frac{r_B}{\sum r + r_S} + 1 + t.$$

Here, t depends on the network condition and the system capability (e.g., CPU, I/O, and memory). Our experience shows that 3-10 seconds for t is a reasonable setting. Assume the server can provide enough bandwidth so that $r_S + \sum r > r_B$, L will be 5-12 seconds.

5.3 Cluster-Aware Prefetching

To achieve fast and smooth transition, we further introduce a cluster-aware prefetching in NetTube. Our experiment on 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

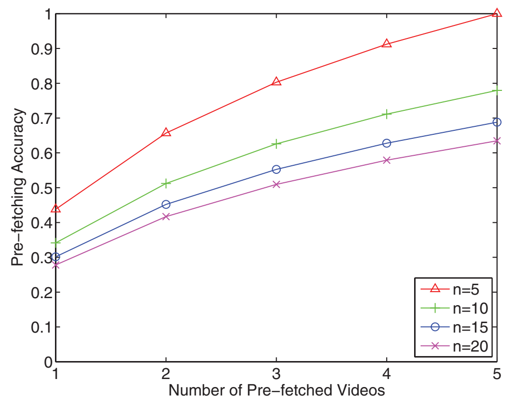

Fig. 10. Prefetching accuracy as a function of the number of prefetches.

average video length is 200 seconds, if we prefetch 10 seconds 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. 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 repositories of movies or TV programs, the next video is 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 $s = \sum_{i=1}^{r}(\frac{1}{i})$. Assume the top p videos are to be prefetched, the probability of a successful hit of the next video is $\sum_{i=1}^{p}(\frac{1}{i \cdot s})$. Figure 10 plots the estimated prefetching accuracy as a function of number of prefetches for different numbers of related videos. With the average number of related videos being 10, prefetching 3 video prefixes can achieve an accuracy of 62%, with a cost of about 1 MByte for typical YouTube videos.

While this prefetching accuracy is not exceptionally high, we emphasize that it is for the prefetching during one video playback. With the existence of video interest correlation, the effectiveness of prefetching can be much higher after a client plays back multiple videos. The intuition is that, given that the videos are generally clustered, the videos prefetched for the current video are likely in the related list of the next video, and so forth. Hence, as long as a client keeps all the prefetched prefixes, which consumes a small disk space, the chance that a prefix to be prefetched has already in its cache will only increase over the time. The client can skip it and instead fetch another video's prefix, and the prefetching accuracy will accordingly increase.

To illustrate this, we now offer a simple analysis. We assume that each time a peer prefetches p distinct video prefixes, so that when the peer starts prefetching after it finishes downloading the i-th videos, it has already 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.[3] Since the newly prefetched 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) is only uniformly distributed in the rest of the list.

[3]Our simulations show that, for the crawled datasets, the average of c is 0.3225 and the median is 0.3061.

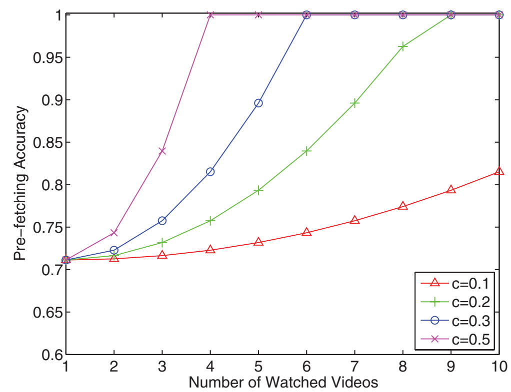

Fig. 11. Prefetching accuracy as a function of the number of watched videos.

Table I. Bandwidth Capacity and Distribution of Users

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

Figure 11 plots the estimated prefetching accuracy as a function of the number of watched videos for different c, where p is set to 4. Clearly, when the number of watched videos increases, the prefetching accuracy increases, and so for larger c. Given the typical c of 0.3, the accuracy exceeds 90% after playback 5 videos.

6. PERFORMANCE EVALUATION

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

6.1 Configuration

Our simulation is based on part of our crawled dataset, which contains $6,951$ distinct videos, each having at least one and up to 20 related videos. This representative subset of the data 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 Huang et al. [2007] as shown in Table I.

The video bitrate is set to 384 kbps, typical for YouTube videos, as well as short videos of other sites [Cheng et al. 2008]. The peer arrival rate follows a Poisson distribution $\frac{\lambda^n \cdot e^{-\lambda}}{n!}$. We change the value of λ from 1 to 8, in order to emulate different client population. We assume that a client randomly selects the first video within a list of predefined popular 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. The duration a peer stays in the system follows a Weibull distribution modeled from our latest measurement on PPVA[4]: $\frac{k}{\lambda'}(\frac{x}{\lambda'})^{k-1}e^{-(x/\lambda')^k}$, where $\lambda' = 108$ and $k = 0.553$. For simplicity, we assume the peer will finish watching each video in its entirety. When the duration of the peer is greater than a randomly assigned lifetime following the above distribution,

[4]www.ppacc.com.

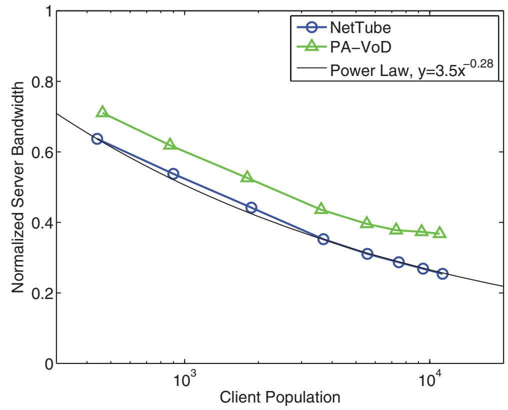

Fig. 12. Server bandwidth consumptions of NetTube and PA-VoD (normalized by the total bandwidth in the client/server model).

the user will leave the system. The given 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.

6.2 Server Workload 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 a client/server system and the Peer-Assisted VoD (PA-VoD) [Huang et al. 2007], 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 the video correlation. More detailed description of PA-VoD can be found in Huang et al. [2007].

Figure 12 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 much 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. As shown in the figure, the trend of the normalized server bandwidth in NetTube can be well modeled by a power law distribution. For 12000 online clients, it requires only 25% of the bandwidth as with client/server, suggesting that NetTube is quite scalable with client population.

Considering the following YouTube statistics: 14.6 billion views per month, 15 page views per user, and 20 minutes staying time per user [Alexa 2010], we simply derive the number of simultaneous online users as about 450 thousand. Hence we get the normalized server bandwidth using NetTube as 0.09. Since YouTube spends $1 million a day to pay for the server bandwidth using the client/server architecture, utilizing NetTube can reduce this expense to about $90 thousand only.

To further understand the savings, we examine the distribution of the downloading sources in the overlays. Figure 13 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 perform similarly. However, when the online clients increase, more and more NetTube peers do not download 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.

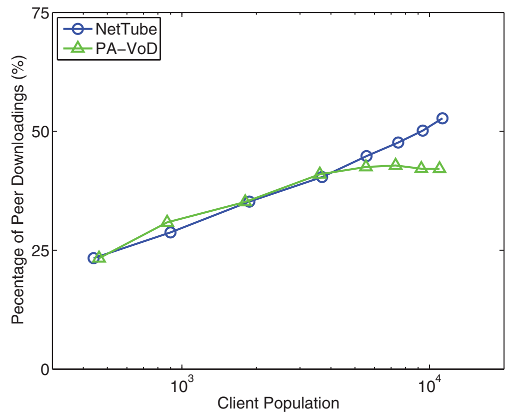

Fig. 13. Percentage of peer downloadings (as opposite to downloadings from the server) in NetTube and PA-VoD.

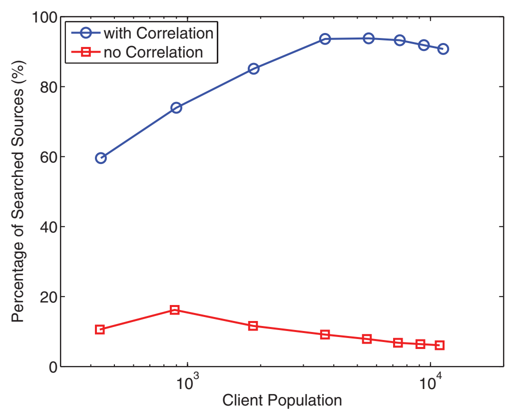

Fig. 14. Percentage of searched sources.

6.3 Impact of Interest Correlation

We next examine how NetTube benefits from the existence of the video interest correlation. To this end, we create a dataset with independent videos yet preserving the popularity distribution of 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 aware of correlation.

For each playback by a client, we record the number of source peers returned by the search among the client's neighbors (see Section 4). We then calculate the percentage of sources returned from the neighbors over the total number of the available sources, that is, all other clients that have cached the video. Figure 14 compares the percentages with and without correlation. We can see that, with video correlation, the percentage increases very fast before the online peers reaches 3000, approaching nearly 95%. It slowly drops as the 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 larger client population. This in turn suggests that the existence of video interest correlation indeed makes our correlation-aware search quite effective, despite its simplicity and limited search

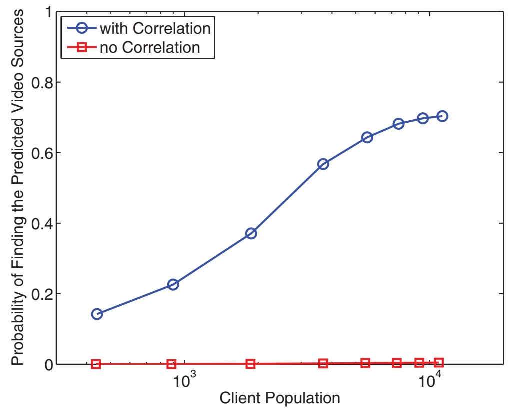

Fig. 15. Probability of neighbors having the predicted video.

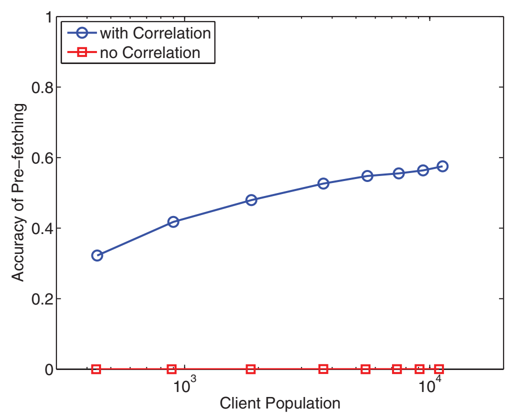

Fig. 16. Accuracy of prefetching.

range (2 hops only). In comparison, if correlation is ignored, peers can only find less than 20% sources, and this percentage keeps decreasing.

We further plot in Figure 15 the probability that a peer's neighbors have the video to be prefetched. We can see that the probability in the simulation with video correlation (0.7 when online peers are more than 7000) is much higher than that without correlation (nearly 0), confirming that with video interest correlation, neighbors are more likely to share similar interests.

Figure 16 compares the accuracy of prefetching in the simulations with and without video interest correlation. The accuracy in the simulation with correlation is about 55%, which is much higher than that in the simulation without correlation (nearly 0). The latter is so low simply because the video repository is too large and a small number of tries (e.g., 10 times) can hardly hit the next video if the videos are independent. In other words, given the sheer 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 prefetching strategy if we are blind to video correlation. We note that the accuracy of NetTube is lower than our analysis in Section 5.3, mainly because when a 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

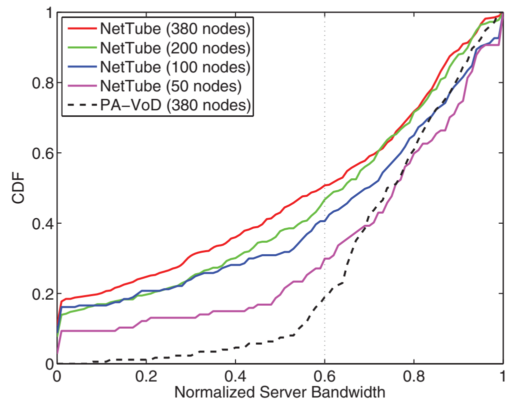

Fig. 17. Cumulative distribution of normalized server traffic.

number of online peers increases, the accuracy increases, because with larger user base, more videos are likely to be downloaded and cached.

7. PROTOTYPE EXPERIMENTS OVER PLANETLAB

To further investigate the performance of NetTube, we have implemented a prototype and conducted experiments over the PlanetLab network.[5] 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. The dataset is a portion of the dataset used in the simulation, and we have confirmed that it has the same property (clustering coefficient and characteristic path length) as the original dataset. We have conducted a series experiment with the number of online nodes ranging from 50 to 380.[6] Other configurations are similar to that in the simulations, except that if a client suffers severe playback delay (60 seconds), it will fail and the client program will restart.[7]

We again look at the bandwidth consumption of the server first. To this end, we record the total downloading traffic and downloading traffic from the server for each client. We plot the cumulative distribution function (CDF) of the normalized server bandwidth in Figure 17. For 380 concurrent clients, near 51% of them have downloaded less than three fifths of the traffic from the server, and the numbers are 47%, 41% and 30% peers for 200, 100 and 50 clients, respectively. In contrast, only 19% peers save that much bandwidth with PA-VoD for 380 nodes. We can naturally expect that the savings of NetTube will be more significant with larger client populations, as validated by our simulation results.

We also record the number of neighbors a peer connected when it finished all the watching, or when the experiment finishes. Since in PA-VoD, a peer joins a new overlay when watch a new video, we calculate the average number of neighbors for each peer. The results are plotted in Figure 18. It is clear that a peer has much more neighbors in NetTube system than in PA-VoD. This confirms that our

[5]http://www.planet-lab.org.

[6]We have filtered out inactive PlanetLab nodes and obtained totally 380 active nodes. To conduct the smaller scale experiment (50, 100, and 200 nodes), we randomly selected the expected number of nodes among them.

[7]To understand the failure, we have examined the individual logs recorded by each node. Most of the failures are caused by the fact that the node cannot receive any data. Yet we traced the senders and found that they had actually sent out data. This implies that the receiver program would have been suspended. Therefore, we added a timer to the program and examined the system clock. We found that there were receivers having the suspension problem, which was caused by system resource shortage at the respective PlanetLab nodes.

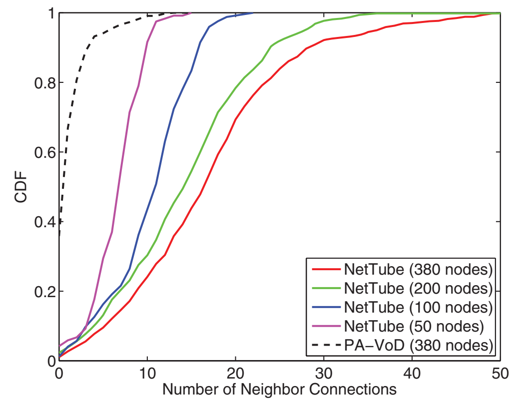

Fig. 18. Cumulative distribution of connected neighbors.

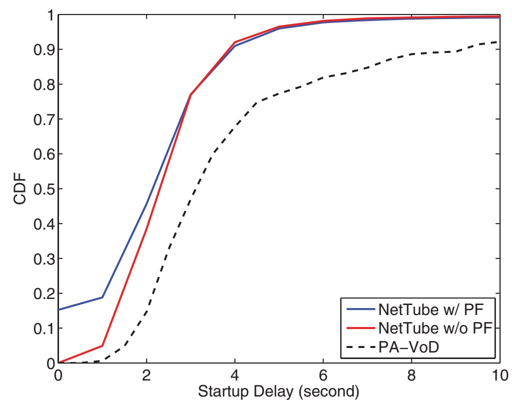

Fig. 19. Cumulative distribution of startup delays.

system uses bilayer overlay to increase the size of overlay, and thus makes peer-to-peer delivery more efficient.

We next examine the startup delay, that is, the interval from selecting a new video to starting playback this video. We assure that a peer can only start to play the video when the playback buffer contains the first second part of the video, thus the startup delay is the time to search the sources and to fill up the one-second playback buffer. We examine NetTube with and without prefetching, and also compare it with the PA-VoD.

We plot the CDF of the startup delay in Figure 19, for the maximum of 380 nodes, and find that NetTube performs much better than PA-VoD. Even without prefetching, more than 90% of the startups are within 4 seconds in NetTube, whereas only 68% can achieve this in PA-VoD. This is because the delay-aware scheduling mechanism prioritizes the peers that are about to encounter delay, including the peers that are starting. In NetTube with prefetching, about 18% of the startups are 0, indicating that the users can seamlessly move to the next video. In summary, the average startup delays for NetTube are 2.8 and 3.0 seconds with and without prefetching, respectively, which is much lower than that of PA-VoD (5.6 seconds). We believe that this is mainly due to the delay-aware scheduling mechanism and cluster-aware prefetching.

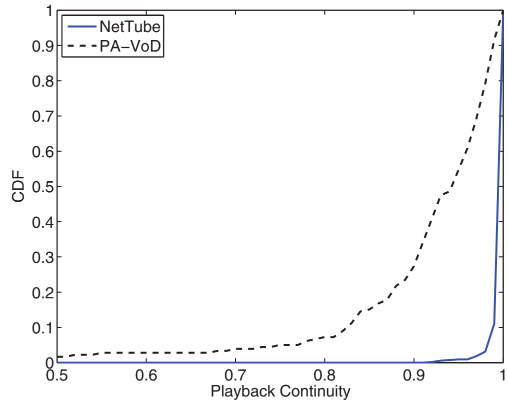

Fig. 20. Cumulative distribution of playback continuity.

We have tried prefetching in PA-VoD, too. Yet, with the extremely low accuracy (as shown in Figure 16), it does not help at all. We also find that the client population does not affect the result of NetTube without pre-fetching and that of PA-VoD; while as the population decreases, the startup delay of NetTube with prefetching approaches to that of NetTube without prefetching. This is because when the client population is low, the probability of the neighbors having the predicted video is low, as shown in the simulation (Figure 15). Nevertheless, it also indicates that, with a larger user base, NetTube with prefetching will perform better.

Finally, we investigate the playback quality experienced by individual clients. We record the summation of all the watched videos' lengths for each peer D_{watch} and the delay it experienced during the playback (D_{delay}, excluding the startup delay). We calculate a *playback continuity* as $\frac{D_{watch}}{D_{watch}+D_{delay}}$. The playback continuity being 1 indicates that the peer did not encounter any delay. In Figure 20, we plot the CDF of the playback continuity, which shows that, in NetTube, 99% of the clients enjoy a continuity over 0.95, and 90% of the clients even reach 0.99 continuity. In contrast, less than 50% of the PA-VoD client can achieve 0.95, and there are about 7% peers suffering from continuity less than 0.8. The results confirm that our NetTube design can better accommodate peer churns introduced by frequent transitions among short videos. They also surpass our previous experimental results without the delay-aware scheduling [Cheng and Liu 2009], demonstrating the superiority of this novel mechanism customized for short videos.

8. CONCLUSION

YouTube-like sites have become one of the killer Internet applications, 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 socialized 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 exhibits strong interest correlation, which does not exist in traditional video sites either.

We then proposed NetTube, a novel peer-to-peer assisted delivering framework leveraging the video interest correlation. We addressed a series of key design issues to realize the system, including a bi-layer overlay, an efficient indexing scheme, a delay-aware scheduling mechanism, and a pre-fetching strategy benefiting from interest correlation. Its performance was evaluated through both simulations and PlanetLab experiments, which demonstrates the necessity and superiority of utilizing video interest correlation.

REFERENCES

AGGARWAL, V., CALDEBANK, R., GOPALAKRISHNAN, V., JANA, R., RAMAKRISHNAN, K., AND YU, F. 2009. The effectiveness of intelligent scheduling for multicast video-on-demand. In *Proceedings of the 17th ACM International Conference on Multimedia (MM)*. ACM, New York, NY, 421–430.

ALBERT, R., JEONG, H., AND BARABÁSI, A.-L. 1999. The diameter of the World Wide Web. *Nature 401*, 130–131.

ALEXA. 2010. Youtube.com site info. http://www.alexa.com/siteinfo/youtube.com.

BENEVENUTO, F., RODRIGUES, T., CHA, M., AND ALMEIDA, V. 2009. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 49–62.

BLOG, O. Y. 2009. Zoinks! 20 hours of video uploaded every minute! http://youtube-global.blogspot.com/2009/05/zoinks-20-hours-of-video-uploaded-every_20.html.

BLOOM, B. 1970. Space/time trade-offs in hash coding with allowable errors. *Comm. ACM 13,* 7, 422–426.

CARTER, L. 2008. Web could collapse as video demand soars. http://www.telegraph.co.uk/news/uknews/1584230/Web-could-collapse-as-video -demand-soars.html.

CHA, M., KWAK, H., RODRIGUEZ, P., AHN, Y.-Y., AND MOON, S. 2007. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 1–14.

CHA, M., RODRIGUEZ, P., CROWCROFT, J., MOON, S., AND AMATRIAIN, X. 2008. Watching television over an IP network. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 71–84.

CHENG, X., DALE, C., AND LIU, J. 2008. Statistics and social network of YouTube videos. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS)*. IEEE, Los Alamitos, CA, 229–238.

CHENG, X. AND LIU, J. 2009. NetTube: Exploring social networks for peer-to-peer short video sharing. In *Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 1152–1160.

CHENG, X., LIU, J., AND DALE, C. 2010. Understanding the characteristics of internet short video sharing: A YouTube-based measurement study. *IEEE Trans. Multimedia*.

CORBETT, C. 2006. Peering of video. http://www.nanog.org/mtg-0606/pdf/bill.norton.3.pdf.

GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. 2007. YouTube traffic characterization: A view from the edge. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 15–28.

GOPALAKRISHNAN, V., BHATTACHARJEEY, B., RAMAKRISHNAN, K., JANA, R., AND SRIVASTAVA, D. 2009. CPM: Adaptive video-on-demand with cooperative peer assists and multicast. In *Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 91–99.

HUANG, C., LI, J., AND ROSS, K. 2007. Can internet video-on-demand be profitable? In *Proceedings of ACM SIGCOMM*. ACM, New York, NY, 133–144.

HUANG, Y., FU, T. Z., CHIU, D.-M., LUI, J. C., AND HUANG, C. 2008. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proceedings of the ACM SIGCOMM*. ACM, New York, NY, 375–388.

LA MONICA, P. R. 2006. Google to buy YouTube for $1.65 billion. http://money.cnn.com/2006/10/09/technology/googleyoutube_deal.

LAI, K. AND WANG, D. 2010. The implication of external links on video sharing sites: Measurement and analysis. In *Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY.

LIANG, C., GUO, Y., AND LIU, Y. 2009. Investigating the scheduling sensitivity of P2P video streaming: An experimental study. *IEEE Trans. Multimedia 11,* 3, 348–360.

LIU, J., RAO, S. G., LI, B., AND ZHANG, H. 2008. Opportunities and challenges of peer-to-peer Internet Video Broadcast. *Proc. IEEE 96,* 1, 11–24.

LIU, Z., WU, C., LI, B., AND ZHAO, S. 2009. Distilling superior peers in large-scale P2P streaming systems. In *Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 82–90.

MAGHAREI, N. AND REJAIE, R. 2009a. Overlay monitoring and repair in swarm-based peer-to-peer streaming. In *Proceedings of the 19th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY, 25–30.

MAGHAREI, N. AND REJAIE, R. 2009b. PRIME: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Trans. Netw. 17,* 4, 1052–1065.

MISLOVE, A., MARCON, M., GUMMADI, K., DRESCHEL, P., AND BHATTACHARJEE, B. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 29–42.

O'REILLY, T. 2005. 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.

QIU, X., WU, C., LIN, X., AND LAU, F. C. 2009. InstantLeap: Fast neighbor discovery in P2P VoD streaming. In *Proceedings of the 19th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY, 19–24.

RAO, L. 2010. comScore: YouTube reaches all-time high of 14.6 billion videos viewed in May. http://techcrunch.com/2010/06/24/comscore-youtube-reaches-all-time-high-of-14-6-billion-videos-viewed-in-may.

SAXENA, M., SHARAN, U., AND FAHMY, S. 2008. Analyzing video services in Web 2.0: A global perspective. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY, 39–44.

SCHNEIDER, F., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. 2009. Understanding Online Social Network Usage from a Network Perspective. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 35–48.

SONG, H. H., CHO, T. W., DAVE, V., ZHANG, Y., AND QIU, L. 2009. Scalable Proximity Estimation and Link Prediction in Online Social Networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 322–335.

VENKATARAMAN, V., YOSHIDA, K., AND FRANCIS, P. 2006. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*. IEEE, Los Alamitos, CA, 2–11.

WANG, F., LIU, J., AND XIONG, Y. 2008. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 1364–1372.

WATTS, D. 1999. *Small Worlds: the Dynamics of Networks Between Order and Randomness*. Princeton University Press, Princeton, NJ.

WU, C., LI, B., AND ZHAO, S. 2008. Multi-channel live P2P streaming: Refocusing on servers. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 1355–1363.

WU, D., LIANG, C., LIU, Y., AND ROSS, K. 2009. View-upload decoupling: A redesign of multi-channel P2P video systems. In *Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Los Alamitos, CA, 2726–2730.

YEN, Y.-W. 2008. YouTube looks for the money clip. http://techland.blogs.fortune.cnn.com/2008/03/25/youtube-looks-for-the-money-clip.

YIN, H., LIU, X., QIU, F., XIA, N., LIN, C., ZHANG, H., SEKAR, V., AND MIN, G. 2009. Inside the bird's Nest: Measurements of large-scale live VoD from the 2008 Olympics. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. ACM, New York, NY, 442–455.

ZHANG, X., LIU, J., LI, B., AND YUM, T.-S. P. 2005. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the 24th IEEE Conference on Computer Communications (INFOCOM)*. Vol. 3. IEEE, Los Alamitos, CA, NJ, 2102–2111.

Received February 2010; revised June 2010, July 2010; accepted July 2010