ON THE CHARACTERISTICS AND ENHANCEMENT OF INTERNET SHORT VIDEO SHARING

by

Xu Cheng

B.Sc., Peking University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2008

APPROVAL

Name:	Xu Cheng
Degree:	Master of Science
Title of thesis:	On the Characteristics and Enhancement of Internet Short Video Sharing
Examining Committee:	Dr. Ke Wang Chair
	Dr. Jiangchuan Liu, Senior Supervisor
	Dr. Jian Pei, Supervisor
	Dr. Jie Liang, SFU Examiner

Date Approved: April 2, 2008

Abstract

In this thesis, using long-term data traces, we present an in-depth and systematic measurement study on the characteristics of YouTube, the most successful site providing a new generation of short video sharing service. We find that YouTube videos have noticeable differences compared with traditional videos, making it difficult to use conventional strategies (e.g., peer-to-peer) to reduce the server workload. However, the social network presented among YouTube videos opens new opportunities. We design a novel social network based peer-to-peer short video sharing system, in which peers are responsible for re-distributing the videos that they have cached. We address a series of key design issues to realize the system, including an efficient indexing scheme, a bi-layer overlay leveraging social networking, a source rate allocation and a pre-fetching strategy to guarantee the playback quality. We perform extensive simulations, which show that the system greatly reduces the server workload and improves the playback quality.

Keywords:

YouTube; video on demand; video sharing; measurement; social network; peer-to-peer

Subject Terms:

Multimedia systems; Internet; Communications network architecture; Social networks

To my family

"No Pain, No Gain"

Acknowledgments

I first thank my senior supervisor Dr. Jiangchuan Liu. His significant enlightenment, guidance and encouragement on my research are invaluable for the success of my M.Sc studies. His insights and suggestions have made the road to completing this thesis smoother. I would not have finished this thesis without him.

I thank my supervisor Dr. Jian Pei and my thesis examiner Dr. Jie Liang, for reviewing this thesis and providing helpful suggestions that helped me to improve the quality of the thesis. I also thank Dr. Ke Wang for taking the time to chair my thesis defense. I would also like to extend my gratitude to the faculty and staff in the School of Computing Science at Simon Fraser University.

I thank my colleagues and friends at Simon Fraser University for their help. I especially thank my colleague Cameron Dale, for our cooperation on the YouTube paper. I thank my colleague Feng Wang and ex-colleague Dr. Dan Wang, for their great assistant and encouragement on my research. I also thank Haiyang Wang, Bin Zhou and Zhengbing Bian for their help on this thesis. My friends not only provided help in my studies but also in my everyday life. I am deeply indebted to them.

Last but certainly not least, I thank my family for their love, care and support: my parents, without you, nothing would happen; my aunt, without you, I would not have today's achievement; my brother, you are always my good model; and my girlfriend, your support makes me stronger. This thesis is dedicated to you all. I love you all!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we first introduce the background of Internet short video sharing service, in particular, YouTube. Then we explain our motivation and summarize our contributions in this thesis. Finally we describe the organization of this thesis.

1.1 Background

1.1.1 YouTube

The recent two years have witnessed an explosion of networked video sharing as a new killer Internet application. Among them, *YouTube* [19] is the most successful one, with more than 100 million videos being watched every day [66]. The success of similar sites (i.e., Tudou [18], the most popular video sharing service in China), and the expensive acquisition of YouTube by Google [40], further confirm the mass market interest. Their great achievement lies in the combination of the content-rich videos and, equally or even more importantly, the establishment of social network. These sites have created a video village on the web, where anyone can be a star, from lip-synching teenage girls to skateboarding dogs. With no doubt, they are changing the content distribution landscape and the popular culture [68].

Established in early 2005, YouTube has become one of the fastest-growing websites, and ranks second[1] in traffic among the websites in the Internet by the survey of Alexa [1]. It has a significant impact on the Internet traffic distribution, yet itself is suffering from severe scalability constraints. According to Alexa, the current speed of YouTube is "Slow"

[1]The traffic rank of YouTube was fourth and third on July and November, 2007, respectively.

(average load time is 3.6 seconds) and is slower than 69% of the surveyed sites. Therefore, understanding the features of YouTube and similar video sharing sites is crucial to network traffic engineering and to sustainable development of these new generation of services.

1.1.2 Social Network and Web 2.0

A *social network* is a social structure made of nodes tied by one or more specific types of interdependency. Social network analysis views social relationships in terms of nodes and ties [57]. The *small-world* phenomenon is the hypothesis that the chain of social acquaintances required to connect one arbitrary person to another arbitrary person anywhere in the world is generally short. The concept was first introduced by Milgram [50] in an small-world experiment known as six degrees of separation, in which, a sample of US individuals were asked to reach a particular target person by passing a message along a chain of acquaintances. The average length of successful chains turned out to be about six separation steps. An electronic small world experiment at Columbia University found that about five to seven degrees of separation are sufficient for connecting any two people through e-mail [36].

A social network service uses techniques to build online social networks for communities of people who share interests and activities. Most of them are web based and provide various ways for users to interact, such as chat, email, video, file sharing, blog, discussion groups and so on [58]. Facebook [7], Flickr [8], LiveJournal [11], MySpace [14], Orkut [15], Windows Live Spaces [12] and YouTube are some of the notable social networking websites.

Most of these social networking websites are based on the technique of *Web 2.0* [53], which is a trend in World Wide Web technology and web design. The Web 2.0 marks the new generation of web-based communities and hosted services such as social networking sites, wikis, blogs and folksonomies, which aim to facilitate creativity, collaboration, and sharing among users [64]. The characteristics of Web 2.0 are: rich user experience, user participation, dynamic content, metadata, web standards, scalability, openness, freedom and collective intelligence [24, 41, 53].

1.1.3 VoD and UGC

Video on demand (VoD) systems allow users to select and watch videos over a network as part of an interactive television system. VoD systems either "stream" content, allowing viewing in real time, or "download" it in which the program is brought in its entirety to a set-top box before viewing starts. Nowadays, the delivery devices refer not only to set-top-boxes but also computers, mobile phones and indeed any system that can receive on-demand audio-visual content over a network [61].

Online videos existed long before YouTube entered the scene. However, uploading videos, managing, sharing and watching them were very cumbersome due to a lack of an easy-to-use integrated platform. More importantly, the videos distributed by traditional media servers and peer-to-peer file downloads like BitTorrent were standalone units of content. Each single video was not connected to other related video clips, for example other episodes of a show that the user had just watched. Also, there was very little in the way of content reviews or ratings.

The new generation of video sharing sites, YouTube and its competitors, have overcome these problems. These new generation video sharing sites are mostly *user generated content* (UGC) based, in which the users are active, participatory and creative. The systems allow content suppliers to upload video effortlessly, and to tag uploaded videos with keywords. Users can easily share videos by mailing links to them, or embedding them on web pages or in blogs. Users can also rate and comment on videos, bringing new social aspects to the viewing of videos. Consequently, popular videos can rise to the top in a very organic fashion. The social network existing in YouTube further enables communities and groups. Videos are no longer independent from each other, and neither are users. This has substantially contributed to the success of YouTube and similar sites.

1.2 Motivation

Our motivation of this thesis consists of two aspects:

Since YouTube and similar sites play a more and more important role in today's multi-media world, and re-shape the way people surf the Internet, it is crucial to understand the characteristics of these sites. This target is essential to network traffic engineering and to sustainable development of these new generation of media servers.

On the other hand, it is well-known that most of these new generation of services are suffering from the serious problems of bandwidth consumption and scalability; the widely adopted *peer-to-peer* (P2P) technique is no doubt the most effective strategy to tackle these problems. Our measurement findings provide us both challenges and potentials to utilizing peer-to-peer to enhance these short video sharing sites. Therefore we expect to address a series of problems and implement peer-to-peer in these short video sharing systems.

1.3 Contribution

We summarize our contributions in this thesis:

- We present an in-depth and systematic measurement study on the characteristics of YouTube videos. We have crawled the YouTube site for a four-month period in early 2007, and have obtained 35 datasets totaling 3,269,030 distinct videos' information. This is the largest dataset so far, and constitutes a significant portion of the entire YouTube video repository[2]. Because most of these videos are accessible from the YouTube homepage in less than 10 clicks, they are generally active and thus representative for measuring the repository. Using this collection of datasets, we find that YouTube videos have noticeably different statistics from traditional streaming videos, especially the video length. There are also new features that have not been examined by previous measurement studies, i.e., the growth trend and active life span.

- We also look closely at the social networking aspect of YouTube, as this is a key driving force toward the success of YouTube and similar sites. 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, and creates opportunities for developing novel caching and peer-to-peer distribution schemes to efficiently deliver videos to end users.

- Our social network findings enlighten us the idea of utilizing peer-to-peer technique. We propose a novel social network based peer-to-peer short video sharing system, in which peers are re-distributing the videos that they have cached. We also address a series of key design issues to realize the system, including an efficient indexing scheme, a bi-layer overlay leveraging social networking, a source rate allocation and a prefetching strategy to guarantee the playback quality. We perform extensive simulations and experiments, which show that the system greatly reduces the workload of server and improves the quality of playback.

[2]A video search for "*" as a wildcard character in YouTube returns 80.2 million videos.

1.4 Thesis Organization

The rest of the thesis is organized as follows. We present an overview of relate work systematically in Chapter 2. In Chapter 3, we first describe our method of gathering information for YouTube videos, then present our measurement and analysis on the characteristics and social networking aspect of YouTube video, at last we discuss the implications of the results, and suggest ways that the YouTube service could be improved. In Chapter 4, we propose our social network based peer-to-peer Internet short video sharing design, in which we address some key design issues, presenting the solutions including an indexing scheme, a bi-layer overlay structure, a source rate allocation mechanism and a pre-fetching strategy. We have performed extensive simulations and experiments, and present the result in Chapter 5. Chapter 6 concludes the thesis, and notes some future directions.

Chapter 2

Related Work

In this chapter, we give an overview of the related work systematically. In general, we present the related work in the following three aspects: measurement study on traditional media servers and new generation of media servers, and peer-to-peer video on demand systems.

2.1 Measurement Study on Traditional Media Servers

There has been a significant research effort into understanding the workloads of traditional media servers, looking at, for example, the video popularity and access locality [20, 23, 30, 59]. The different aspects of media and web objects, and those of live and stored video streams have also been compared [31, 60].

In some recent studies, Yu et al. modeled user behavior and relevant access patterns in VoD system [69], and found that the user accesses are predictable but not follow the Poisson distribution. Cheng et al. studied a P2P VoD system GridCast, finding that the key factor of satisfactory user experience is the popularity of channels and a moderate number of concurrent users [29]. Huang et al. analyzed a 9-month trace of MSN Video [13], Microsoft's VoD service, examining the user behavior and popularity distribution of videos [45], and the analysis led to a peer-assisted VoD design for reducing the server's workload.

Our Difference from Theirs

We have found that, while sharing similar features, many of the video statistics of these traditional media servers are quite different from YouTube, i.e., the video length and active life span. More importantly, these traditional studies lack a social networking among the videos.

2.2 Measurement Study on New Generation Media Servers

We have seen simultaneous works investigating popular Web 2.0 sites these two years. The authors in [43] were the first to study the social networking aspect in YouTube. Mislove et al. studied four online social networking sites, including Flickr, YouTube, LiveJournal and Orkut [51], and confirm the power-law, small-world and scale-free properties of online social network. Cha et al. studied YouTube and Daum UCC [6], the most popular UGC service in Korea [28], and also proposed some improvement for UGC design. A YouTube traffic analysis is presented in [39], which tracks YouTube transactions in a campus network, and focused on deriving video access patterns from the network edge perspective. In [54], Paolillo found the social core appearing in YouTube. Zink et al. obtained the trace of YouTube traffic and investigated the caching problem [71].

Our Difference from Theirs

Different from [43, 51, 54], we investigate the social networking among the videos instead of users. Since the viewers are not required to register and upload video, the social networking among the user has less impact than that among the videos, and the social networking among the videos provides us the opportunity for improving the system. Our work complements works in [28, 39, 71] by crawling a much larger set of the videos and thus being able to accurately measure their global properties, in particular, the social network, which they did not consider in their works.

2.3 Peer-to-Peer Video On Demand

There are fewer practical P2P VoD systems than P2P live streaming system (e.g., Cool-Streaming [5, 70] and PPLive [17]). GridCast [9] is one of the P2P VoD system deployed on CERNET (China Education and Research Network). Joost [10] is another new P2P VoD created by the founder of Skype and Kazaa. However, there are numerous literatures on peer-to-peer video on demand.

Most of the works consider multicast, in which the system constructs multicast overlays and peers send data to their downstream peers. Cui et al. proposed application-layer multicast, which takes advantage of the strong buffering capabilities of end hosts in application-layer overlay network [34], and proposed an overlay multicast strategy called oStream to address the on-demand media distribution problem in overlay network. Guo et al. extended the IP multicast patching scheme by proposing P2Cast [42], which constructs the application overlay appropriate for streaming and provides continuous stream playback in the face of disruption from an early departing client. Do et al. presented a system called P2VoD [35], and introduced the concept of generation, which allow clients join the system fast and recover failures in a quick and localized manner.

Several other problems have been addressed as well. For example, in [47], Liao et al. proposed a scheduling scheme for P2P VoD. In [45, 46], Huang et al. proposed a peer-assisted VoD design for reducing the server's bandwidth costs. Mol et al. addressed the problem of free-rider, who downloads data but uploads little or no data in return [52]. In a recent study [38], Garbacki et al. investigated the video piece-selection strategy and use entropy as a measure in a peer-to-peer VoD system, and also introduced helpers to provide more bandwidth.

Our Difference from Theirs

All of the previous works except the last one considered the traditional VoD, and most of them utilized overlay multicast. However, for new generation VoD, the technique has to be substantially revised. The concept of peer-assisted in [45, 46] is close to ours. However, MSN Video is a more traditional video service, with much fewer videos, most of which are longer than YouTube videos, and MSN Video also has no listings of related videos or user information, thus no social networking aspect. Therefore, their approach is not suitable for short video sharing system. In [38], despite they used YouTube videos as experiment data, they did not emphasize this feature, and thus their approach has to be revised as well.

More importantly, to the best of our knowledge, there is no study considering the social networking in video on demand system. This aspect is probably the most important contribution in this thesis. Our measurement study in Chapter 3 reveal the small-world property among the YouTube videos, and we thus propose a novel social network based peer-to-peer system in Chapter 4.

Chapter 3

Measurement Study of YouTube

In this chapter, we present a measurement study of YouTube, today's most popular new generation of VoD website. We first briefly introduce YouTube's technology and describe our methodology for measurement, then present the characteristic of YouTube. We also study the social network of YouTube videos, as this is a key driving force toward the success of YouTube. Furthermore, we discuss the implications of the measurement results, briefly suggesting ways that the YouTube service could be improved.

3.1 YouTube Primer

3.1.1 Video Techniques

YouTube video playback technology is based on Adobe's Flash Player 9 and uses the Sorenson Spark H.263 video codec with pixel dimensions of 320 by 240. This technology allows YouTube to display videos with quality comparable to more established video playback technologies (such as Windows Media Player, RealPlayer and QuickTime). YouTube officially accepts uploaded videos in .WMV, .AVI, .MOV and .MPG formats, which are converted into .FLV (Adobe Flash Video) format after uploading [67]. It has been recognized that the use of a uniform easily-playable format has been a key in the success of YouTube.

Each YouTube video is accompanied by the full HTML markup for embedding it within another page, unless the uploader chooses to disable this embedding feature. This simple cut-and-paste feature is especially popular with users of social networking sites, and is a key in the success of YouTube as well.

3.1.2 Video Meta-data

YouTube assigns each video a distinct 11-digit ID composed of 0-9, a-z, A-Z, - and _. Each video contains a series of meta-data: video ID, uploader, date when it was uploaded, category, length, user ratings, number of views, ratings and comments, and a list of "related videos". The related videos are links to other videos that have a similar title, description or tags, all of which are chosen by the uploader. A video can have hundreds of related videos, but the webpage only shows at most 20 at once. A typical example of the meta-data is shown in Table 3.1.

ID	ELhtXtnV3pg
Uploader	Alkarin
Date Added	May 19, 2007
Category	Entertainment
Video Length	268 seconds
Number of Views	596,272
Rate	4.83
Number of Ratings	1,227
Number of Comments	1,475
Related Videos	aUXoekeDIW8, 30MBljXxg3M, 4_Ow2wTuSHE, Sog2k6s7xVQ, ...

Table 3.1: Meta-data of a YouTube video

3.2 Methodology and Dataset Description

We have crawled the YouTube site for a four-month period and obtained information on its videos through a combination of the YouTube API and scrapes of YouTube video web pages. The results offer a series of representative partial snapshots of the YouTube video repository as well as its growth trends.

We consider all the YouTube videos to form a directed graph, where each video is a node in the graph. If a video b is in the related video list (first 20 only) of a video a, then there is a directed edge from a to b. Our crawler uses a breadth-first search to find videos in the graph. We define the initial set of 0-depth video IDs, which the crawler reads in to a queue at the beginning of the crawl. When processing each video, it checks the list of related videos and adds any new ones to the queue. Given a video ID, the crawler first extracts information from the YouTube API [2], which contains all the meta-data except date added, category and related videos. The crawler then scrapes the video's webpage to obtain the remaining information.

Our first crawl was on February 22nd, 2007, and started with the initial set of videos from the list of "Recently Featured", "Most Viewed", "Top Rated" and "Most Discussed", for "Today", "This Week", "This Month" and "All Time", which totalled 189 unique videos on that day. The crawl went to more than four depths (crawled to fifth depth but did not finish), finding approximately 750 thousand videos in about five days. In the following weeks we ran the crawler every two to three days, each time defining the initial set of videos from the list of "Most Viewed", "Top Rated", and "Most Discussed", for "Today" and "This Week", which is about 200 to 300 videos. On average, the crawl finds 73 thousand distinct videos each time in less than 9 hours.

To study the growth trend of the video popularity, we also use the crawler to update the statistics of some previously found videos. For this crawl we only retrieve the number of views for relatively new videos (uploaded after February 15th, 2007). This crawl is performed once a week from March 5th to April 16th 2007, which results in seven datasets.

We also separately crawled the file size and bit-rate information. To get the file size, the crawler retrieves the response information from the server when requesting to download the video file and extracts the information on the size of the download. Some videos also have the bit-rate embedded in the FLV video meta-data, which the crawler extracts after downloading the meta-data of the video file.

Finally, we have collected the information about YouTube users. The crawler retrieves information on the number of uploaded videos and friends of each user from the YouTube API, for a total of more than 1 million users.

3.3 Characteristic of YouTube

From February 22nd to May 18th, 2007, we have obtained 35 datasets totaling 3,269,030 distinct videos. In the measurements, some characteristics are static and can be measured once from the entire dataset (e.g., category, length, and date added), some are dynamic and can change from dataset to dataset (e.g., number of views). We consider this dynamic information to be static over a single crawl. Later, the updated number of views information will be used to measure the growth trend and active life span. Finally, we show the result of user behaviors, including number of uploads and number of friends.

3.3.1 Video Category

One of 12 categories is selected by the user when uploading the video. Table 3.2 lists the number and percentage of all the categories, which is also shown graphically in Figure 3.1. In our entire dataset we can see that the distribution is highly skewed: the most popular category is "Music", at about 22.9%; the second is "Entertainment", at about 17.8%; and the third is "Comedy", at about 12.1%.

Category	**Count**	**Percentage**
Autos & Vehicles	83818	2.6%
Comedy	395064	12.1%
Entertainment	582249	17.8%
Film & Animation	272581	8.3%
Gadgets & Games	240290	7.4%
Howto & DIY	65560	2.0%
Music	747562	22.9%
News & Politics	144332	4.4%
People & Blogs	245159	7.5%
Pets & Animals	63159	1.9%
Sports	311374	9.5%
Travel & Places	70970	2.2%
Unavailable	29637	0.9%
Removed	17275	0.5%

Table 3.2: List of YouTube video categories

In the table, we also list two other categories. "Unavailable" are videos set to private, or videos that have been flagged as inappropriate video, which the crawler can only get information for from the YouTube API. "Removed" are videos that have been deleted by the uploader, or by a YouTube moderator (due to the violation of the terms of use), but still are linked to by other videos.

On November, 2007, YouTube updates its category options: "Travel & Places" becomes "Travel & Events", "Howto & DIY" becomes "Howto & Style", "Gadgets & Games" becomes part of "Entertainment". It also adds three new categories: "Education", "Nonprofits & Activism" and "Science & Technology". Then, "Music" category has been broken down into genres such as "Rock", "Pop", "Rap & Hip-Hop", "Classical" and more, making it

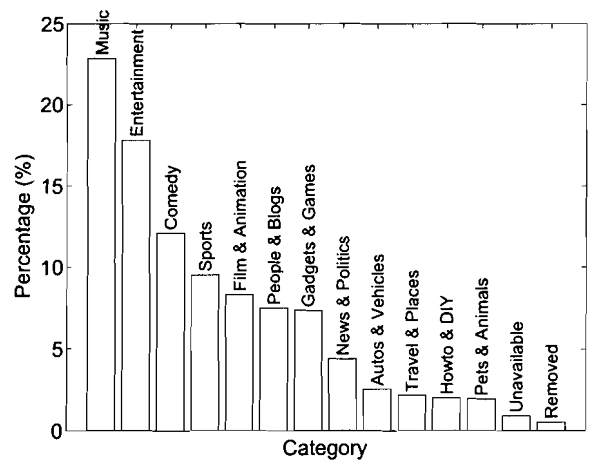


Figure 3.1: Distribution of YouTube video categories

easier to find music videos [4]. However, the data we crawled are prior these updates, so we only analysis the old categories.

3.3.2 Date Added – Growth Trend of Uploading

A study in July, 2006, showed that YouTube have 65,000 daily uploading. During our crawl we record the date that each video uploaded. Because we do not crawl all the YouTube video data, our crawled data cannot indicate the exact number of uploading, but we can study the growth trend of YouTube uploading, which reflects the growth trend of YouTube website. Figure 3.2 shows the histogram of number of new videos added every week in our entire crawled dataset.

Our first crawl was on February 22nd, 2007, so we can get the early videos only if they are still existing or are linked to by other videos we crawled. February 15th, 2005 is the day that YouTube was established, and we can see there is a slow start, as the earliest video we crawled was uploaded on April 27th, 2005. After 6 months from YouTube's establishment, the number of uploaded videos increases steeply. We use a power law curve to fit this trend.

Note that in the dataset we collected, the number of uploaded videos decreases steeply starting in March, 2007. However, this does not imply that the uploading rate of YouTube videos has suddenly decreased. The reason is that many recently uploaded videos have not been so popular, and are probably not in other videos related videos' list. Since few videos link to those new videos, they are not likely to be found by our crawler. Nevertheless, as

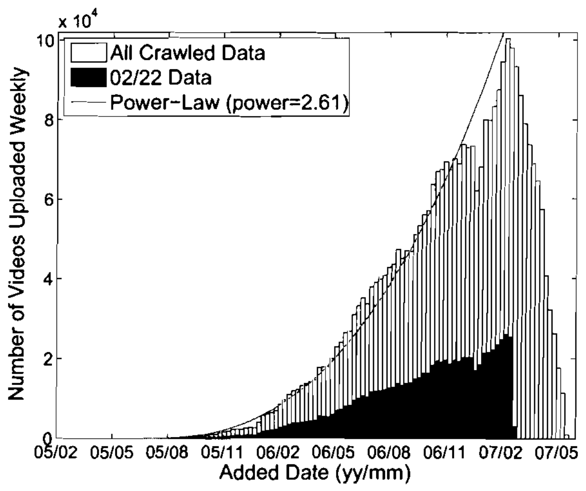


Figure 3.2: Histogram of YouTube videos uploaded weekly among crawled data

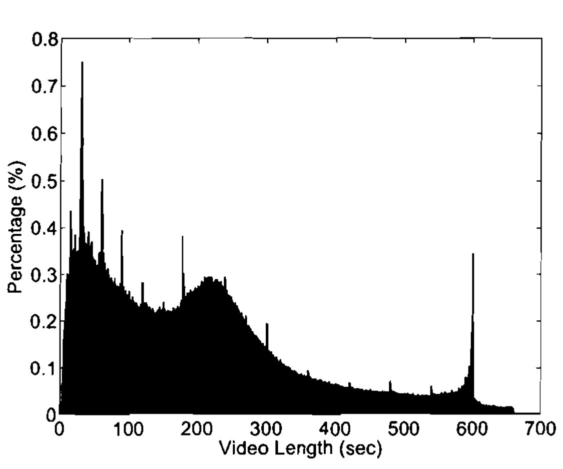
those videos become popular or get linked by others, our crawler may find them and get their information. Comparing the entire dataset to the first and largest dataset, which was crawled on February 22nd, we also see the same trend.

3.3.3 Video Length

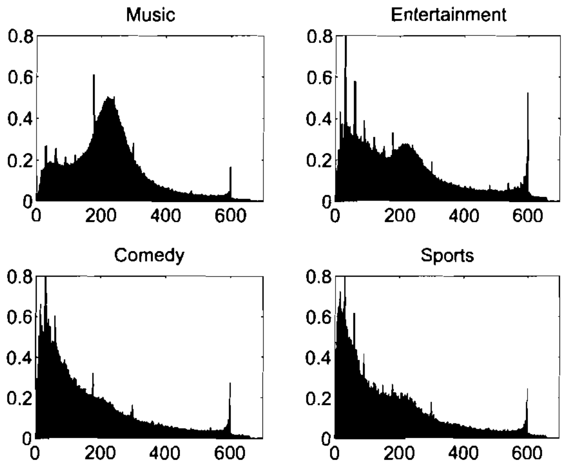
The length of YouTube videos is the most distinguished difference from traditional media content servers. Most traditional servers contain a small to medium number of long videos, typically 1-2 hour movies (e.g., HPLabs Media Server [59]), whereas YouTube is mostly comprised of videos that are short clips.

In our entire crawled dataset, 97.9% of the videos' lengths are within 600 seconds, and 99.1% are within 700 seconds. This is mainly due to the limit of 10 minutes imposed by YouTube on regular users uploads. We do find videos longer than this limit though, as the limit was only established in March, 2006, and also the YouTube Director Program allows a small group of authorized users to upload videos longer than 10 minutes [4].

Figure 3.3a shows the distribution of YouTube videos' lengths within 700 seconds, which exhibits three main peaks. The first peak is within one minute, and contains more than 20.6% 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 17.1% of the videos. This peak is mainly caused by the large number of videos in the "Music" category, since "Music" is the most popular category for YouTube, and the typical length of a "Music" video is often



(a) All crawled YouTube video



(b) The four most popular categories

Figure 3.3: Distribution of YouTube video length

within this range (shown in Figure 3.3b). The third peak is near 10 minutes; it is caused by the limit on the length of uploaded videos, which encourages some users to circumvent the length restriction by dividing long videos into several parts, each being near the limit of 10 minutes. We can also observe that there are peaks at around every exact minute.

Figure 3.3b shows the video length distributions for the top four most popular categories. We can see "Music" videos have a very large peak between three and four minutes (about 27.6%) due to the typical music video length as mentioned above. "Entertainment" videos have the greatest peak at around 10 minutes comparing to other categories, because most of these videos are talk shows, which are half hour to several hours in length typically, but have been cut into several parts near 10 minutes. In comparison, "Comedy" and "Sports" videos have much more videos within two minutes (about 50.9% and 50.4% respectively), probably corresponding to "highlight" type clips.

3.3.4 File Size and Bitrate

Using video IDs from a normal crawl, we retrieved the file size of about 184 thousand videos. Not surprisingly, we find that the distribution of video file sizes is very similar to the distribution of video lengths. We plot the cumulative distribution function (CDF) of YouTube video file size in Figure 3.4.

We find that in our crawled data, 98.3% of the videos are less than 25 MB, and the

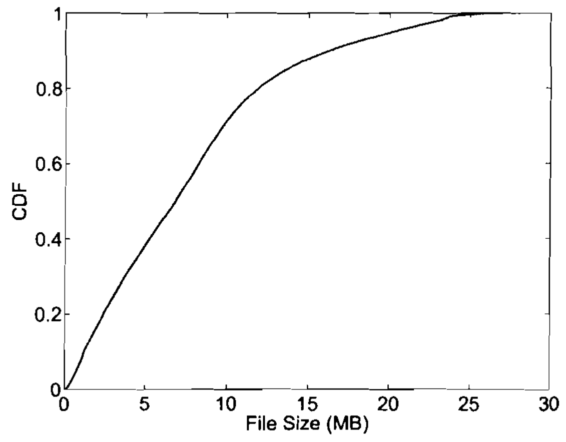


Figure 3.4: CDF of YouTube video file size

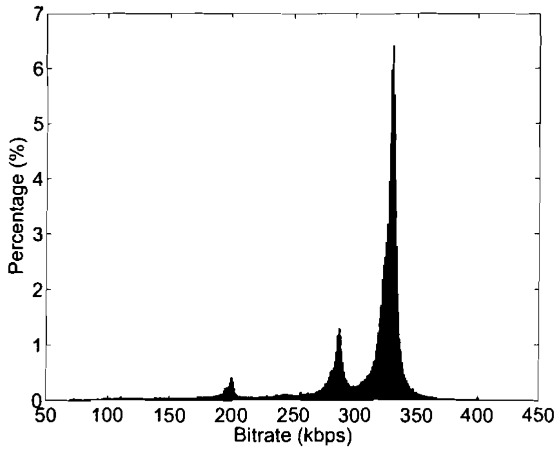


Figure 3.5: Distribution of YouTube video bitrate

average size is 8.4 MB, which is quite small comparing to hundreds MB movies. However, considering there are over 80.2 million YouTube videos, the total disk space required to store all the videos is more than 650 TB! Smart storage management is thus quite demanding for such an ultra-huge and still growing site, which we will discuss in more detail in Section 3.5.

We found that 87.6% of the videos we crawled contained FLV meta-data specifying the video's bitrate in the beginning of the file, indicating that they are constant-bitrate (CBR) videos. For the rest of the videos that do not contain this meta-data (probably variable-bitrate, or VBR, videos), we calculate the average bit-rate from the file sizes and their lengths. In Figure 3.5, we observe that the videos' bitrate has three clear peaks. Most videos have a bitrate around 330 kbps (70.6% video bitrate are between 300 kbps and 360 kbps), with two other peaks at around 285 kbps and 200 kbps. This implies that YouTube videos have a moderate bitrate that balances quality and bandwidth.

3.3.5 Views – User Access Pattern

The number of views a video has had is the most important characteristic we measured, as it reflects the popularity and access patterns of the videos. Because this property is changing over time, we cannot use the entire dataset that combines all the data together. Therefore we use a single dataset from April 3rd, 2007, containing more than 100 thousand videos, which is considered to be relatively static.

Figure 3.6 shows the number of views as a function of the rank of the video by its

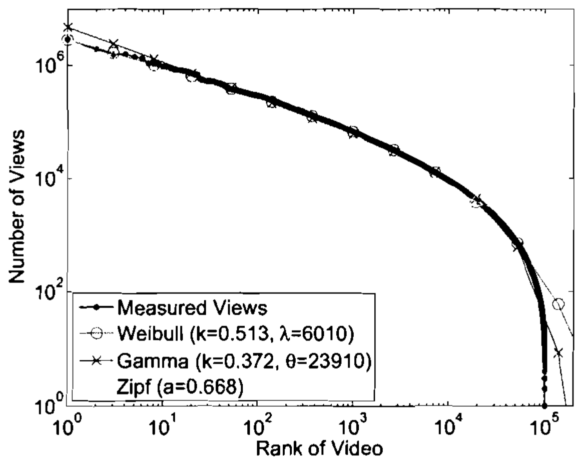


Figure 3.6: YouTube videos rank ordered by popularity

popularity. Though the plot has a long tail on the linear scale, it does NOT follow a Zipf distribution, which should be a straight line on a log-log scale. This is consistent with some previous observations [20, 23, 59, 69] that also found that video accesses on a media server does not follow Zipf's law. We can see in the figure, the beginning of the curve is linear, but the tail (after 2×10^3 videos) decreases tremendously, indicating there are not so many less popular videos as Zipf's law predicts.

One reason to explain this feature is probably that users are likely to access their own videos several times after uploading it to make sure the videos are uploaded successfully, although the videos will possibly be never accessed again. Moreover, different from visiting webpage, as a user will probably visit one page many times, a user is not likely to watch the same video many times, which leads the distribution not to follow the Zipf's law.

This result seems consistent with some results [69], which also has a heavy tail. However, it differs from others [20, 23, 59], in which the curve is skewed from linear from beginning to end. Their results indicate that the popular videos are also not as popular as Zipf's law predicts, which is not the case in our measurement. To fit the skewed curve, some use a concatenation of two Zipf distributions [23], while others use a generalized Zipf distribution [59]. Because our curve is different, we attempted to use two different distributions, Weibull and Gamma, with a Zipf to compare with. We find that both distributions fit better than Zipf, due to the drop-off in the tail (in log-log scale) that they have.

Using the seven datasets of updated views, we can calculate the incremental views of the videos for different span of time, as Figure 3.7 shown. The downmost (red) plot is the incremental views for one week, and the upmost (blue) plot is the incremental views for six weeks. We can see as the time passes, the curve has a more and more heavy tail.

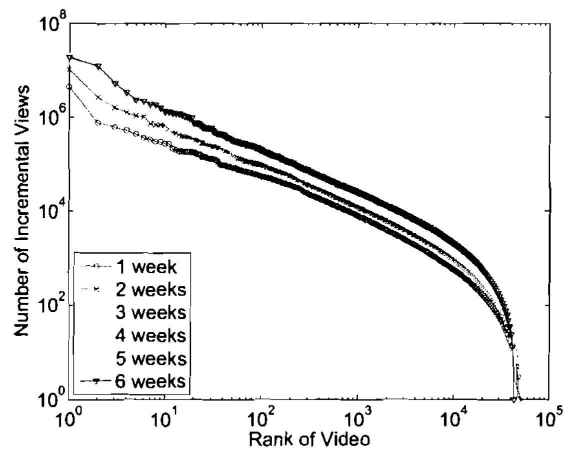


Figure 3.7: YouTube videos rank ordered by incremental views

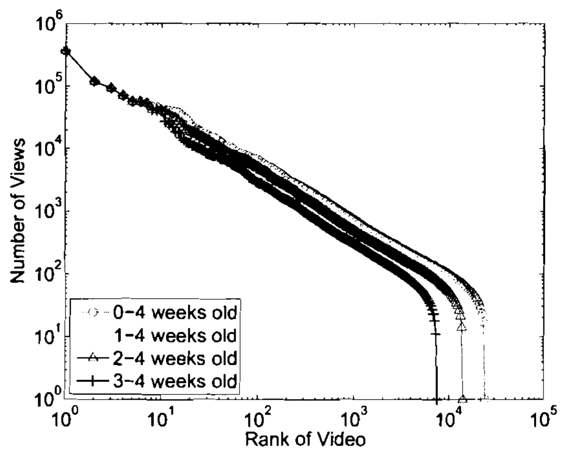


Figure 3.8: Recently added YouTube videos rank by popularity

Validation of Unbiased Sample Data

We were initially concerned that the crawled data might be biased, as popular videos may be more likely to appear in our BFS crawl than non-popular ones. Since the whole video name space is too large (2^{66}), random sampling directly can be quite difficult. Therefore, we have been saving the recently added videos from the YouTube RSS feed for four weeks, as sampling from these is close to random. We update the views counts of these videos, and plot in Figure 3.8. The left-most (black) plot is only the videos added during the first week (all the 3-4 weeks old videos), while the right-most (red) plot contains all the videos (all the 0-4 weeks old videos). There is a clear tail in all the plots, verifying that our BFS search does find non-popular videos just as well as it finds popular ones.

Correlation Between Length and Views

We next investigate the correlation between video's length and number of views. We divide the dataset into three groups and calculate the statistics of views. Table 3.3 lists the fact of these statistics. We can see that longer videos are slightly more popular than very short videos, and the deviations in all of the three groups are very large. Therefore we can claim that the correlation is not strong.

length	count	min	max	mean	median	std
$\leq$ 100s	32316	0	1922218	3475	448	21420
100s–240s	34817	0	1381993	5159	820	24447
$>$ 240s	34213	0	2839893	5599	1057	28132

Table 3.3: Correlation between video's length and number of views

Correlation Between User Rating and Views

YouTube enable registered user to rate a video according to his or her preference. We study the correlation between user rating and number of views, as we will study the number of ratings in Section 3.3.7. The ratings shown on the webpage are the number of red stars. However, the ratings we crawled are float numbers, which are the average scores. Table 3.4 lists the fact of the statistics of views in different ratings. We can see that high rating videos are more popular than low rating videos, but the deviation is larger as well.

rating	count	min	max	mean	median	std
0–1	24204	0	795127	373	119	5491
1–2	1910	5	479183	3053	884	15293
2–3	6424	8	794806	3300	752	18928
3–4	11633	4	1922218	4907	1126	28046
4–5	57175	3	2839893	6828	1486	29405

Table 3.4: Correlation between user rating and number of views

Correlation Between Age and Views

At last we examine the correlation between age of video and number of views. We calculate the statistics in Table 3.5. Not surprisingly, the video age clearly affects the number of views, because older videos have more opportunity to be accessed.

However, we can see in the younger video group that there are very popular videos, and in the older video group there are very unpopular videos. In fact, the deviations in all the groups are quite large. This indicates that different videos have different growth trends, making popular video more popular, and unpopular video less popular.

age	count	min	max	mean	median	std
$\leq$ 1 month	22662	0	1556837	2244	226	17852
1–3 month	22939	0	1922218	2450	467	17279
3–6 month	24980	0	934062	4163	903	18639
6–12 month	27570	3	1051432	7670	1912	26165
$>$ 12 month	3195	0	2839893	19095	4598	79142

Table 3.5: Correlation between video age and number of views

3.3.6 Growth Trend of Views and Active Life Span

From above, we know that some are very popular (their number of views grows very fast), while others are not. Also, after a certain period of time, some videos are almost never watched. Starting on March 5th, 2007, we updated the number of views statistic of relatively new videos (uploaded after February 15th, 2007) every week for seven weeks. To be sure the growth trend will be properly modeled, we eliminate any videos that have been removed and so do not have the full seven data points, resulting in a dataset size totaling approximately 43 thousand videos.

We have found that the growth trend can be modeled better by a power law than a linear fit. Therefore, a video can have a increasing growth trend (if the power is greater than 1), a constant growth trend (power near 1), or a slowing growth trend (power less than 1). The trend depends on the exponent used in the power law, which we call the growth trend factor p. We model the views count after x weeks as

$$v(x) = v_0 \times \frac{(x+\mu)^p}{\mu^p} \tag{3.1}$$

where μ is the number of weeks before March 5th that the video has been uploaded, x indicates the week of the crawled data (from 0 to 6), and v_0 is the number of views the video had on March 5th.

We modeled the 43 thousand videos using Equation 3.1 to get the distribution of growth trend factors p, which is shown in Figure 3.9. 70% of the videos have a growth trend factor that is less than 1, indicating that most videos grow more and more slowly as time passes.

Since YouTube has no policy on removing videos after a period of time or when their popularity declines, the life span of a YouTube video is almost infinite. However, the video's popularity may grow more and more slowly, and will almost stop growing after some time, which we define as the video's *active life span*. From this active life span, we can extract the feature of a video's temporal locality.

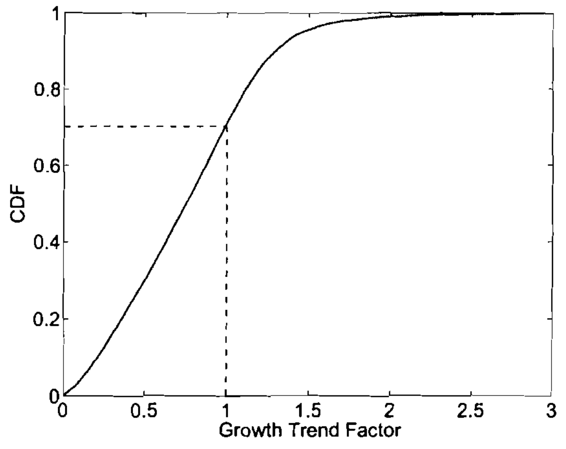


Figure 3.9: Distribution of YouTube video growth trend factor

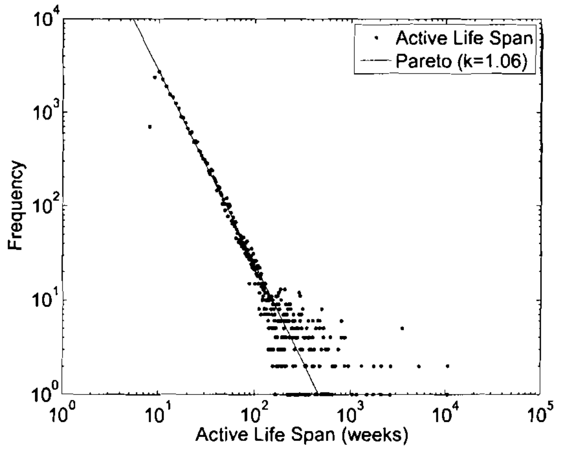


Figure 3.10: Distribution of estimated active life span for t=10%

If a video's number of views increases by a factor less than t from the previous week, we define the video's active life span to be over. We prefer this relative comparison to an absolute comparison, since we are only concerned with the shape of the curve instead of the scale. For each video that has a growth trend factor p less than 1, we can compute its active life span l from

$$\frac{v(l)}{v(l-1)} - 1 = t \tag{3.2}$$

which can be solved for the active life span

$$l = \frac{1}{\sqrt[p]{1+t} - 1} + 1 - \mu \tag{3.3}$$

Thus we see that the active life span is dependent on the growth trend factor p and the number of weeks the video has been on YouTube, but does not depend on the number of views the video had at the start of the experiment.

Figure 3.10 shows the probability density function (PDF) for the active life span of the approximately 30 thousand videos (with p less than 1), for a life span factor of $t = 10\%$. The solid line is the Pareto distribution fit to the data, which fits very well, and results in a parameter k of 1.06. From looking at multiple fits with various values of t, we find that they all result in the same parameter k, the only difference is the location of the line.

Since we do not have the server logs of YouTube, we cannot accurately measure a video's temporal locality, which would show whether recently accessed videos are likely to be accessed in the near future. However, the active life span gives us another way to view the temporal locality of YouTube videos. Figure 3.10 implies that most videos have a short active life span, which means the videos have been watched frequently in a short span of time. Then, after the video's active life span is complete, fewer and fewer user will access them. This characteristic has good implications for web caching and server storage. We can design a predictor to forecast the active life span using our model from Equation 3.3, which can help a proxy or server to make more intelligent decisions, such as when to drop a video from the cache. We will discuss this in more detail in Section 3.5.

3.3.7 User Behavior

User behavior is also an important feature that worths studying. We first study the characteristic of number of ratings and comments from the same dataset as number of views, and then examine the user statistics from another dataset including the number of uploaded videos and the number of friends.

Ratings and Comments

Whenever a video webpage has been accessed, the number of views increases, whereas users need to log in to rate and comment so that the number of ratings and comments will increase. Therefore ratings and comments better reflect the user behavior.

Figure 3.11 plots the number of ratings against the rank by the number of ratings, and similarly for the comments. The two have a similar distribution, and we note that the tails do not drop so quickly as that of number of views. We list the statistics of ratings and comments in Table 3.6, along with views for comparison.

	min	**max**	**mean**	**median**	**std**	**zeros**
views	0	2839893	4771	741	24892	0.1%
ratings	0	34401	14.5	3	128	21.6%
comments	0	3899	8.7	2	39	33.6%

Table 3.6: Facts of views, ratings and comments

We can see the comments are fewer than that of ratings, and both are much fewer than views. In fact, a great amount of videos do not have a rating or a comment. This indicates that users are more willing to watch videos rather than to log in to rate and make comments.

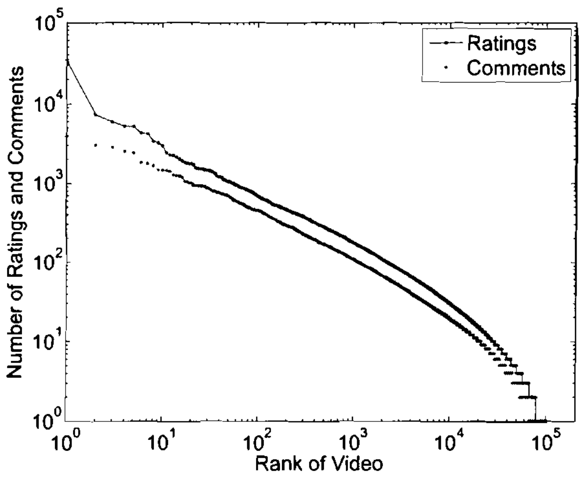


Figure 3.11: YouTube videos ratings and comments ranks by the numbers of ratings and comments

User Uploads and Friends

YouTube currently has about 2,830,000 registered users.[1] Users need to login to upload video or watch some limited videos. A user can add another one to the friend list so that it is convenience to watch friends' video. From the crawl of user information we performed on May 28th, 2007, we can extract two characteristics of YouTube users: the number of uploaded videos and the number of friends.[2] We did this for more than 1 million users found by our crawler in all the crawls performed before this one.

YouTube is a typical UGC website, in that all the videos are uploaded by the users. These active users are also a key of the success of YouTube. Figure 3.12 plot the distribution of user uploaded videos' count. We can see there are many users have uploaded a few videos, and a small number of users have uploaded a great number of videos. Since we collect the user ID from the previous crawled data, all of these user have uploaded at least one videos. However, the information of users that do not upload video has not been crawled. We believe that there are a great number of users have not uploaded a single video.

We plot the distribution of number of friends each user has in Figure 3.13, as well as

[1] A channel search for "*" as a wildcard character in YouTube returns about 3.79 million channels, of which a registered user has one.

[2] We also collect the information of number of watched videos. However, this information is not representative because users are not required to login to surf YouTube. Hence this statistic may not correctly indicate the actual number of watched videos for each user.

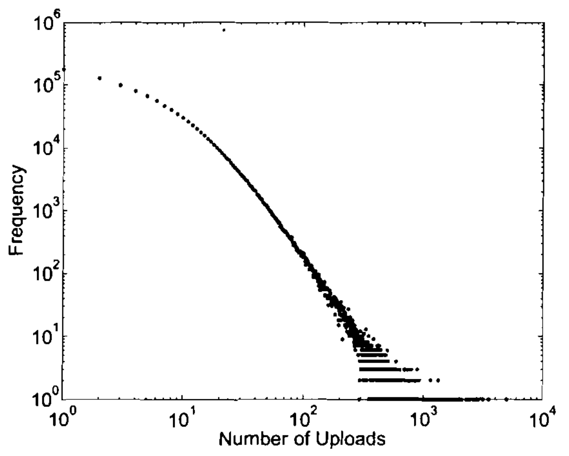


Figure 3.12: Distribution of user uploaded videos' count

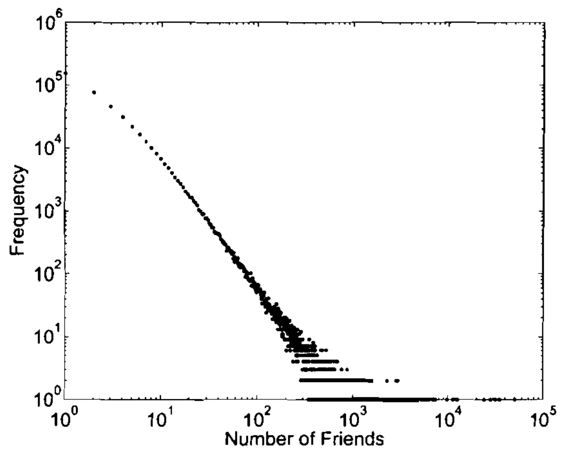


Figure 3.13: Distribution of number of user friends

min	max	mean	median	std	zeros
0	50614	4.3	0	96	58%

Table 3.7: Statistics of users' friends count

the statistics in Table 3.7. Interestingly, in over 1 million users data, we have found that 58% of the users have no friends. Therefore, we can see that having friend does not affect the access pattern, so that the correlation between users is not very strong. Thus the social networking existing among users has less impact than that among videos, as we will study in the next section.

3.4 Social Network of YouTube

YouTube is a prominent social media application. There exist communities and groups in YouTube, there are statistics and awards for videos and personal channels, and so videos are no longer independent from each other. It is therefore important to understand the social networking characteristics of YouTube. We next examine the social network among YouTube videos, which is a very unique and interesting aspect of this kind of video sharing sites, as compared to traditional media services.

3.4.1 Small-World Phenomenon

Small-world network phenomenon is probably the most interesting characteristic for social networks. The concept was first introduced by Milgram [50] to refer to the principle that people are linked to all others by short chains of acquaintances (AKA six degrees of separation). It has been found in various real-world situations: URL links in the Web [22], Gnutella's search overlay topology [48], and Freenet's file distribution network [44].

Watts and Strogatz describe networks that are neither completely random, nor completely regular, but possess characteristics of both [63], and introduce a measure of one of these characteristics, the cliquishness of a typical neighborhood, as the *clustering coefficient* of the graph. They define a small-world graph as one in which the clustering coefficient is still large, as in regular graphs, but the measure of the average distance between nodes (the *characteristic path length*) is small, as in random graphs.

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. For a node of degree k_i, the maximum possible number of directed edges between neighbors of the node is $k_i(k_i - 1)$; for undirected graphs, half as many are possible. The clustering coefficient of the graph $C(G)$ is then the average of the clustering coefficients of all nodes in the graph. Watts and Strogatz show that many real-world networks exhibit small-world behavior [63]. They calculate the clustering coefficient for some networks to be: 1 for cliques, 0.79 for the network of collaborating actors in Hollywood, a maximum of 0.75 for circulant graphs, 0.28 for the neural network of Caenorhabditis elegans, and almost 0 for random networks.

Given a graph $G = (V, E)$, the characteristic path length d_i of a node $i \in V$ is the average of the minimum number of hops it takes to reach all other nodes in V from node i. The characteristic path length of the graph $D(G)$ is then the average of the characteristic path length of all nodes in the graph. The characteristic path length is only defined, in the case of directed graphs, if the graph is strongly connected. Otherwise, since there exists at least one node which can not be reached from another node, the characteristic path length is infinite. The graph formed by URL links in the world wide web has a characteristic path length of 18.59 [22].

3.4.2 The Small-World in YouTube

We have obtained the cumulative datasets, each consisting of all the previous crawled data, so the last one contains all the data we crawled. We select the first one, the last one, and other eight cumulative datasets, such that the increment between each consecutive two datasets is similar (about 270 thousand). We measured the graph topology by using the related links in YouTube pages to form directed edges in a video graph for each dataset. Figure 3.14 shows an example of a dataset graph, containing 2940 nodes. Videos that have no outgoing or no incoming links are removed from the analysis.

Figure 3.14: An example of a dataset graph

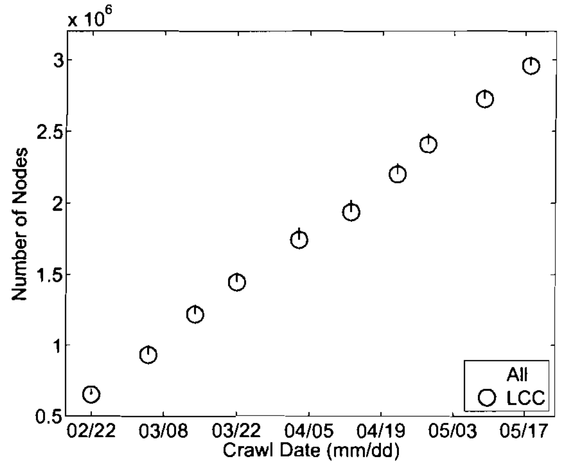


Figure 3.15: Size of cumulative datasets

Since not all of YouTube is crawled, the resulting graphs are not strongly connected, making it difficult to calculate the characteristic path length. Therefore, we also use the *Largest strongly Connected Component* (LCC) of each graph for the measurements. For comparison, we also generate random graphs that are strongly connected. Each of the random graphs has the same number of nodes and average node degree of the LCC of the crawled dataset, and is also limited to a maximum node out-degree of 20, similar to the crawled datasets. Figure 3.15 shows the sizes of ten cumulative datasets and their LCC graphs, which are very close to the total dataset sizes, suggesting that each large dataset is connected strongly.

Figure 3.16 shows the clustering coefficient for the entire graph, as a function of the size of the dataset. The clustering coefficient is quite high (around 0.29), especially in comparison to the random graphs (nearly 0). There is a slow drop in the clustering coefficient for larger datasets, showing that there is some inverse dependence on the size of the graph, which is common for some small-world networks [55].

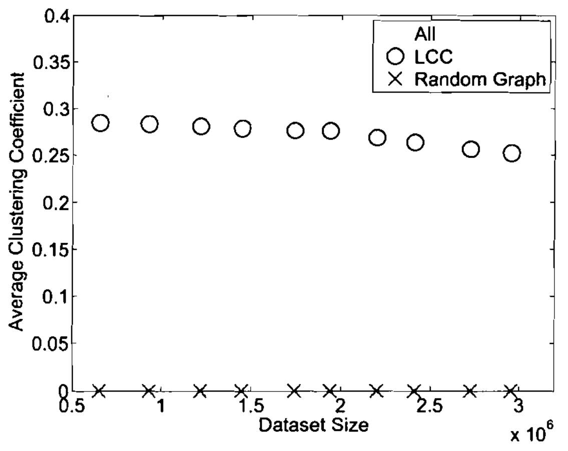


Figure 3.16: Clustering coefficient of cumulative datasets

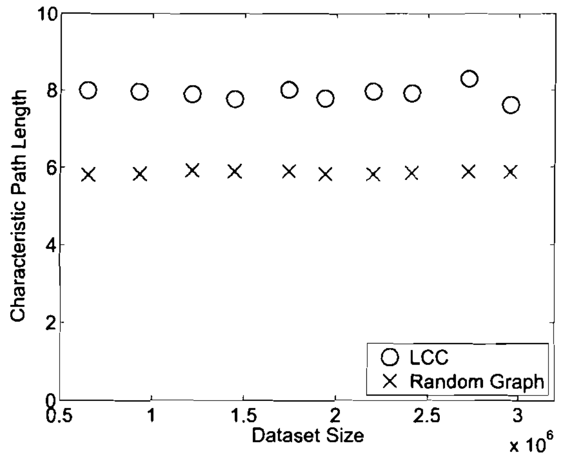


Figure 3.17: Characteristic path length of cumulative datasets

Figure 3.17 shows the characteristic path length for the graphs. We can see that the average diameter (about 8) is only slightly larger than that of a random graph (about 6), which is quite good considering the still large clustering coefficient of these datasets.

The network formed by YouTube's related videos list has definite small-world characteristics. The clustering coefficient is much larger than a similar sized random graph, while the characteristic path lengths are approaching the shortest path lengths measured in the random graphs. This finding is expected, due to the user-generated nature of the tags, title and description of the videos that is used by YouTube to find related ones.

These results are similar to other real-world user-generated graphs that exist, yet their parameters can be quite different. For example, the graph formed by URL links in the world wide web exhibits a much longer characteristic path length of 18.59 [22]. This could possibly be due to the larger number of nodes (8×10^8 in the web), but it may also indicate that the YouTube network of videos is a much closer group.

3.5 Discussion

A study in June 2007 shows that YouTube alone has comprised approximately 20% of all HTTP traffic, or nearly 10% of all traffic on the Internet [37]. An official report in June

2006 reveals the YouTube's outbound bandwidth is about 20 Gbps with a 20% growth rate per month [32]. Assuming the network traffic cost is $10 per Mbps, the estimated YouTube transit expense is currently more than $11 million per month. This high and rising expense for network traffic is probably one of the reasons YouTube was sold to Google [40].

According to Alexa, the current speed of YouTube is "Slow" (average load time is 3.6 seconds) and is considered slower than 69% of the surveyed sites. Scalability is no doubt the biggest challenge that YouTube faces, particularly considering that websites such as YouTube survive by attracting more users. In this section, we briefly discuss the implications of our measurement results toward improving the scalability of YouTube.

3.5.1 Implications on Proxy Caching and Storage Management

Caching frequently used data at proxies close to clients is an effective way to save backbone bandwidth and prevent users from experiencing excessive access delays. Numerous algorithms have been developed for caching web objects or streaming videos [49]. While we believe that YouTube will benefit from proxy caching, three distinct features call for novel cache designs: first, the number of YouTube videos (80.2 million) is orders of magnitude higher than that of traditional video streams (e.g., HPC: 2999, HPL: 412 [59]); second, the size of YouTube videos is also much smaller than a traditional video (a YouTube video of average of 8.4 MB versus a typical MPEG movie of about 700 MB); finally, the views rank of YouTube videos do not well fit a Zipf distribution, which has important implications on web caching [26].

Considering these factors, full-object caching for web or segment caching for streaming video are not practical solutions for YouTube, whereas prefix caching [56] is probably the best choice. Suppose for each video, the proxy will cache a 5 second initial clip (about 200 KB) of the video. Given the Weibull distribution of view frequency suggested by our measurements, assuming that the cache space is devoted only to the most popular videos, we calculate and plot the hit-ratio as a function of the cache size in Figure 3.18. To achieve a 90% hit-ratio, the proxy would require about 6 GB of space for the current YouTube video repository. Such demand is acceptable for today's proxy servers.

The cache efficiency can be further improved by exploring the small-world characteristic of the related video links (see Section 3.4.2). That is, if a group of videos have a tight relation, then a user is likely to watch another video within the group after finishing the first one. This expectation is confirmed by Figure 3.19, which shows a clear correlation

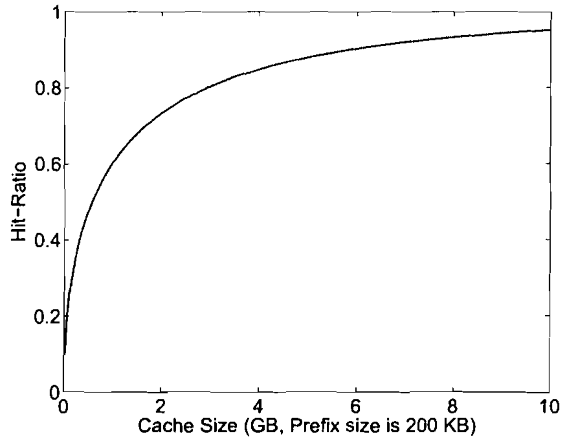


Figure 3.18: Prefix caching hit-ratio as a function of cache size

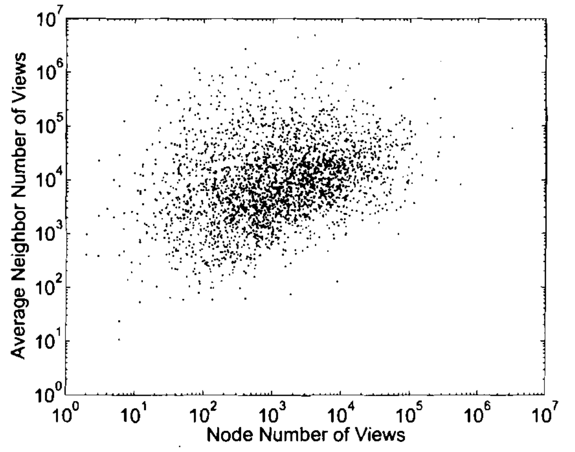


Figure 3.19: Correlation of average neighbor views and views

between the number of views and the average of the neighbors' number of views. Once a video is played and cached, the prefixes of its directly related videos can also be prefetched and cached, if the cache space allows.

Given the constant evolution of YouTube's video repository, a remaining critical issue is when to release the space for a cached prefix. We found in Section 3.3.6 that the active life span of YouTube videos follows a Pareto distribution, implying that most videos are popular during a relatively short span of time. Therefore, a predictor can be developed to forecast the active life span of a video. With the predictor, the proxy can decide which videos have already passed their life span, and replace it if the cache space is insufficient.

The life span predictor can also facilitate disk space management on the YouTube server. Currently, videos on YouTube servers will not be removed by the operator unless they violate the terms of service. With a daily 65,000 new videos introduced [66], the server storage will soon become a problem. A hierarchical storage structure can be built with videos passing their active life span being moved to slower and cheaper storage media.

3.5.2 Can Peer-to-Peer Save YouTube?

Short video sharing and *peer-to-peer* (P2P) streaming have been widely cited as two key driving forces to Internet video distribution; yet their development remains largely separated. The P2P technology has been quite successful in supporting large-scale live video streaming (e.g., CoolStreaming [5, 70] and PPLive [17]). Since each peer contributes its bandwidth to serve others, a P2P overlay scales extremely well with larger user bases.

YouTube and other similar sites still use the traditional *Client/Server* (C/S) architecture, thus restricting their scalability. Therefore implementing peer-to-peer in short video sharing system is quite promising. In the next chapter, we will discuss how to utilize P2P in short video sharing system, and present the design of our social network based P2P short video sharing system.

3.6 Summary

This chapter has presented a detailed investigation of the characteristics of YouTube videos. Through examining the massive amounts of data collected in a four-month period, we have demonstrated that, while sharing certain similar features with traditional video repositories, YouTube exhibits many unique characteristics, especially in length distribution. These characteristics introduce novel challenges and opportunities for optimizing the performance of short video sharing services.

We have also investigated the social networking among YouTube videos, which is probably the most unique and interesting aspect, and has substantially contributed to the success of this new generation of service. We have found that the networks of related videos, which are chosen based on user-generated content, have both small-world characteristics of a large clustering coefficient indicating the grouping of videos, and a short characteristic path length linking any two videos. We have suggested that these features can be exploited to facilitate the design of novel caching and peer-to-peer strategies for short video sharing. In the next chapter, we will propose our design of a social network based peer-to-peer short video sharing system.

Chapter 4

Social Network Based P2P Short Video Sharing

In this chapter, we propose a novel peer-to-peer short video sharing system, which explores the social network. We first analyze the the challenges and potentials to utilize peer-to-peer in short video sharing system such as YouTube. Then we present solutions to a series of key designs issues in this system, including a bi-layer overlay design, an indexing scheme, a source rate allocation mechanism and a pre-fetching strategy.

4.1 Challenges and Potentials

4.1.1 Challenges

Unfortunately, our YouTube measurement results in the previous chapter suggest that utilizing peer-to-peer delivery for YouTube can be quite challenging. Specifically, if we directly construct overlay for each single video, we will encounter following problems:

1. The length of a YouTube video is very short (refer to Section 3.3.3), as many are even shorter than the initialization time of a P2P overlay. Therefore, if we construct overlay for each single video, the peer-to-peer overlay will be too dynamic;

2. Users often frequently load another video when finishing one, and thus the overlay will suffer from an extremely high churn rate. It will increase the initialization overhead;

3. There are a huge number of videos (80.2 million[1]), and the concurrent video views are very few, even at the peak time (5 concurrent views [38]). Therefore, each peer-to-peer overlay will appears very small, thus increasing the overhead of overlay maintenance.

These features will make peer-to-peer in such system inefficient or even infeasible.

4.1.2 Potentials

However, our social network finding could be exploited by considering a group of related videos together. The videos watched by a user are likely to have similar features, and users that are watching the same video are thus likely to have similar preferences, which means the user will be likely to watch another video that has been watched by the other user who is watching the same video.

Consider the peer has cached the downloaded videos and can upload any of them. Since peers are storing the cached videos as long as they exist in the system, the overlay would be much larger and more stable. This behavior can significantly reduce the bandwidth consumption from the server and greatly increase the scalability of the system.

Notice that we will use two terms, "user" and "peer". "User" better indicates the human viewer, and "peer" better indicates the client in the network. Some peer behaviors are not controlled by the users, i.e., uploading and searching, and thus we do not need to consider the problem of leeches or free-riders [52].

4.2 System Overview

In our enhancement, we expect to reduce the server workload, but we do not abandon the server, as we still consider the server as a powerful peer that stores all the entire video files and can provide much uploading bandwidth. In our system, the server initially stores all the video files, and send them to the peers. As more and more videos have been distributed among peers, these peers can be the sources and the server acts as a backup source.

In the previous chapter, we find social networking existing among YouTube videos, which exhibit the small-world properties. This suggests that within a group of videos, if one

[1]This number has been increasing dramatically, as the number is 42.5 on May, 2007, and 71.6 on January, 2008.

video has been watched, another video within the group will be more likely to be accessed immediately than a video outside the group.

Our design is based on the following principle:

peers are responsible for re-distributing the videos that they have cached.

Suppose in an ideal system, peers have unlimited uploading bandwidth and storage size, and will never leave the system once joined. As the peers download more and more videos, the server will need to send fewer and fewer, since these videos are available among the peers. Considering no new videos introduced, after a period of time, a great number of videos have been downloaded, and thus the server will no longer need to send any video, since peers can act as the sources with enough uploading bandwidth. However, this ideal system is impossible to achieve.

In our design, when a new peer joins the system, it first requests a video. If the requested video is not cached by any of the existing peers, the server will send the video file to the peer. Otherwise, peers that have cached that video will send the file, and the server will assure the minimum source rate if needed.

In the following, we present a series of solutions addressing some key design issues: a novel bi-layer streaming overlay exploring social network in Section 4.3, an efficient indexing scheme in Section 4.4, a source rate allocation mechanism assuring the continuous playback in Section 4.5, a pre-fetching strategy reducing the startup delay in Section 4.6.

4.3 Bi-layer Overlay

We propose a novel bi-layer overlay, consisting lower-layer overlays which are per-video overlays for peer downloading and uploading, and upper-layer overlays which are social network based, and benefit the searching for sources. Figure 4.1 illustrates an example of a bi-layer overlay.

4.3.1 The Lower-layer Overlay and Selecting First Video

The peers that are downloading or have downloaded the same video form a *lower-layer overlay.* For example in Figure 4.1, there are five lower-layer overlays: v_1-, v_2-, v_3-, v_4- and v_5-overlay; in v_1-overlay, P_1 and P_2 are sources, sending video file v_1 to P_3 and P_4.

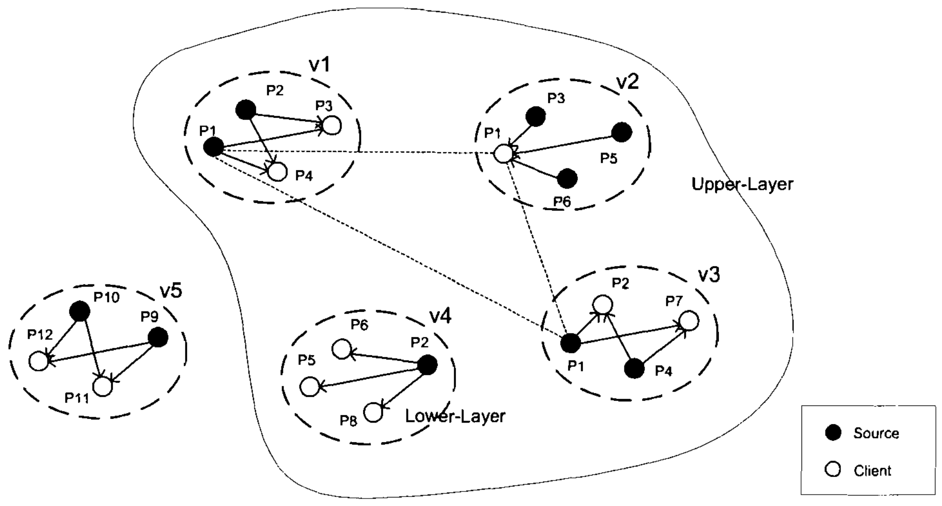


Figure 4.1: Illustration of a bi-layer overlay

Different from traditional VoD system, in which peer are exchanging the video file that they are currently watching, in this system, all the cached videos can also be sources. The authors in [38] used a flashcrowd model to estimate the number of concurrent video views, and found that an average of 5 views during the flashcrowd and of 0.4 views outside of the flashcrowd. This indicates that the number of the peers that are downloading the same video is quite small. Moreover, in short video sharing system, the small length makes the overlay extreme dynamic. An advantage of our design is that it increases the size of overlay and makes the overlay more stable.

Therefore, a peer can appear in several lower-layer overlays. For example, P_1 is caching v_1 and v_3, so it is a source in v_1- and v_3-overlay. In the meantime, it is downloading v_2, so it is also a client in v_2-overlay. We thus define the *neighbors* of a peer P as all the peers that have downloaded from or uploaded to peer P. For example, the neighbors of peer P_1 are P_3, P_4 (in v_1-overlay), P_5, P_6 (in v_2-overlay), and P_2, P_7 (in v_3-overlay). A peer knows which videos have been cached by its neighbors, and when it caches new video or drop any video, it will notify its neighbors and the server.

When a new peer joins the system, it first sends a request for the first video to the server. The server looks up the information of which peers have cached this video and return a list of source peers caching this video. If there is no peer storing this video, which means this peer is the first one to download this video (or some other peers that have downloaded this video but have dropped due to the limited cache size), the server then establishes the connection and sends this video file to this peer. Otherwise, the peer selects the source peers among the list to join the lower-layer overlay, and starts to receive the requested video file from those sources. If the total source rate of the sources is lower than a required minimum rate such that the continuous playback cannot be assured, the server then acts as another source to provide the remaining required streaming rate. A detailed downloading request and source rate allocation mechanism is presented in Section 4.5.

When a peer finishes watching a video and begins to download the next one, it joins another overlay, but still stays in the previous one as a source. If the peer leaves the system, it will leave all the overlays.

Since the cache size is limited, if the cache is full, the peer should drop one cached video. For simplicity, we assume that the cache can store n videos (we do not consider the size of the video). The peer uses the simple first-in-first-out (FIFO) strategy for caching updating. If the peer drops a cached video, say v_o, it will leave the v_0-overlay.

4.3.2 The Upper-layer Overlay and Selecting Next Video

Considering each lower-layer overlay as a "node", these nodes then form an *upper-layer overlay*. The "nodes" in the upper-layer overlay are linked by the same peer appearing in different lower-layer overlays. Therefore, this upper-layer overlay is more conceptional, as videos watched by one peer are likely to be close to each other based on the social network.

For example, in Figure 4.1, peer P_1 exists in v_1-, v_2- and v_3-overlay, so "node" v_1, v_2 and v_3 are connected to form the upper-layer overlay. "Node" v_4 is also in this upper-layer overlay, because peer P_2, P_5 and P_6 are in some other lower-layer overlays. However, "node" v_5 is not in this upper-layer overlay, because none of the peers that are downloading or have downloaded video v_5 appears in other lower-layer overlays.

It is possible that all of the lower-layer overlays are connected together, which means there is no island, because it is possible that peers would watch a completely different and irrelevant video after the previous one. However, the closer the videos are, the more links there will be between the "nodes".

When a peer finishes watching one video and selects another video to download, it first looks up the information of all neighbors, which is stored locally, and adds the neighbor peers that have cached the target video to a peer list. Then it requests for the target video to all its neighbors. The neighbor peers look up their neighbors' information, which are also stored locally, and return their neighbors' address if they have cached that video.

Therefore, the request is in fact a two-hop query flooding. We define a threshold of the maximum number of the peers in the list. If the number does not reach the threshold, the peer then send request to the server to get more source peers. After collecting the sources' information, the peer sends the downloading requests to those sources, as doing so for the first video. Again, the server acts as another source if the source peers cannot satisfy source rate requirement.

The benefit of an upper-layer overlay is that, peers in different lower-layer overlays may be potential neighbors based on the social network. For example, suppose P_4 finishes watching v_1 and prepares for downloading 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, and thus P_4 can then contact with P_3, P_5 and P_6. It is social networking that brings those peers closer.

In our approach of selecting the next video, we use a two-hop flooding. The query flooding can benefit from the social network, as based on the social network, peers downloading the similar videos are close to each other, and thus a small number of hops of flooding will probably find most of the peers that have cached the target video, so that it can offload the server further. However, decreasing the redundancy of the flooding requires a better flooding mechanism, as we list this task as a future work in Chapter 5.

4.4 Indexing Scheme

In this system, we require an indexing scheme, so that the server and peers need to keep track of the all peers' or neighbors' cached videos, and can efficiently find the sources of a video. There are two naive approaches to achieve this, to index in terms of video and to index in terms of peer.

In the first approach, for each video, the system stores the information of all the peers (i.e., IP address) that have cached this video. It is easy to find all the peers that have cached a certain video. However, it is inefficient to manage. For example, when a peer leaves the system without any notification, it is difficult to update the indexing table because the system has to check all the videos to remove that peer. Also the total size is large if the entire video repository is large (i.e., 80.2 million videos in YouTube). Whereas in the second one, for each peer, the system stores the information of all the videos (i.e. video ID) that this peer has cached. It is easy to find out which videos a certain peer has cached and is efficient to manage, but it is inefficient to find the peers that have cached a certain video.

Comparing the two approaches, the first one is probably more suitable for our purpose, because the prompt response time of requesting for a list of peers that have cached a certain video is required in our system. However, the total number of videos is often too large, resulting in a huge size of the indexes. Therefore we propose a space-efficient Bloom filter index, and a fast-searching indexing table. Before we present our indexing scheme, we first give a brief introduction of Bloom filter.

4.4.1 Bloom Filter

Bloom filter, introduced by Burton Bloom in 1970 [25], is a simple space-efficient probabilistic data structure for representing a set of elements, and has been widely adopted for different purposes such as distributed caching, P2P/overlay networks and resource routing.

A Bloom filter is an array of m bits, which are set to 0 initially, to represent a set $S = \{x_1, x_2, \ldots, x_n\}$ of n element. A Bloom filter uses k independent hash functions $h_1, \ldots, h_k$ with range $\{1, \ldots, m\}$. For each element $x \in S$, the bit $h_i(x)$ of the array is set to 1 for $1 \leq i \leq k$. To check if an element y is in S, we need to check if all the positions of $h_i(y)$ are set to 1. If not, then y is clearly not in S. Otherwise, we assume that y is in S with some probability. Therefore, false positives are possible in Bloom filter, which means it assumes that an element is in S but actually not. Note that false negatives are not possible. The probability that a specific bit is 1 is $p = 1 - (1 - \frac{1}{m})^{kn} \approx \mathrm{e}^{-kn/m}$, and the probability of false positive is $f = (1 - \mathrm{e}^{-kn/m})^k$ [27].

4.4.2 Index and Indexing Table

We utilize a Bloom filter index to achieve the efficiency. Each peer uses an m-bit Bloom filter with k hash functions to index its n cached videos. The server works as a tracker, storing all peers' indexes and their addresses in an indexing table described later, and peers also keep track of all the neighbors' indexes and their addresses.

A peer's index is initially set to zero. Every time when the peer finishes downloading a video, the video ID is mapped to k positions with the k hash functions, and those k positions of the m-bit array are set to 1. Note that it is difficult to delete a video index, because it cannot guarantee that the mapped positions of the deleted video do not contain other cached videos' index. Therefore, if the peer needs to remove a cached video due to the limited size, the index of the peer requires re-calculated for each cached video. Since the number of cached video is often quite small, it does not affect the efficiency much. Figure 4.2 illustrates the Bloom filter index of a peer.

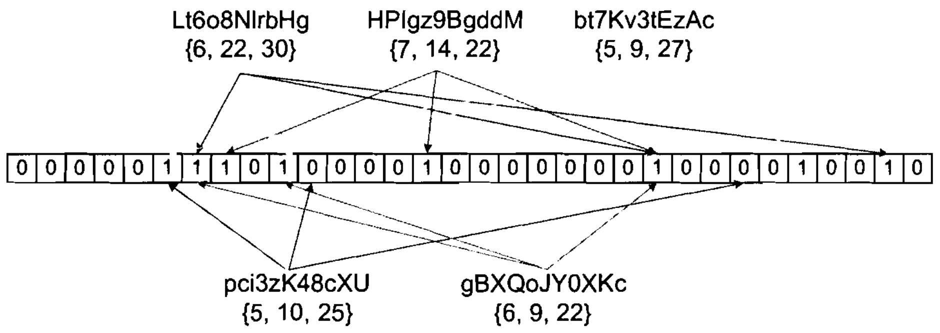


Figure 4.2: Illustration of a peer's index with $m = 32, k = 3$

In the figure, the video "Lt6o8NIrbHg" is mapped to three positions: 6, 22 and 30, and similar to the other two videos. When receives a request for any of these three videos, the peer maps the ID to three positions and checks, and easily finds that it has the video. If receives a request for video "pci3zK48cXU", the peer first maps the ID to three positions: 5, 10 and 25, and then it checks its index, finding that not every position is 1, so it can be certain that the peer does not cached this video. However, if receives a request for video "gBXQoJY0XKc", the peer maps the ID to three positions: 6, 9 and 22, and then finds that all of the three positions are set to 1, so it assume it has the video but actually not.

We design a data structure to organize a list of peer indexes, called *indexing table*, which is shown in Figure 4.3. The server has such a table to track all the peers' indexes, and each peer also has the table to store all its neighbors' indexes. The description of the structure is as follows:

1. The table consists of a head line and N peer lines, where N is the number of online peers;

2. Each peer line has a cell of the peer's IP address, a cell of the integer number of the peer's index (a m-bit integers) and m cells of pointers. For each pointer, if the bit of that position of the index is 0, the pointer points to NULL; if the bit is 1, the pointer points to the next peers whose index bit of that position is also 1. Note that the pointer points to the address (beginning) of the peer line. In the figure, we plot the

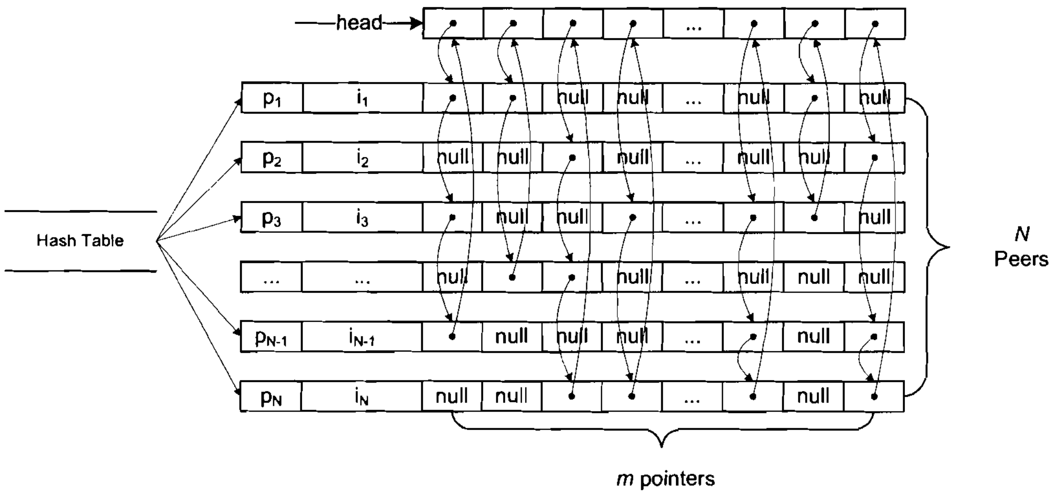


Figure 4.3: Illustration of a indexing table

pointer pointing to the cell in order not to mess the figure. And also, the links are not necessarily like warp and woof, as it depends on the time sequence of when the peer adding the video;

3. The head line has m cells of pointers, pointing to the first peer whose index bit of that position is 1, and the pointer of the last peer whose index bit of that position is 1 points to the head line;

4. A hash table organizes the N peers.

We can see the total size of the table is independent on the total number of videos, and thus is scalable to the number of videos. For each peer line, the size is $8 + m/8 + m \times 4$. Considering $m = 256$, each peer line is about 1064 B. Suppose there are $N = 1 \times 10^4$ online peers, the total size of structure stored in the server is about 10 MB, which is quite acceptable for the server.

4.4.3 Indexing Table Operations

There are five main operations on indexing table: peer joining, peer leaving, peer adding new video, peer deleting video, and searching for all peers having a certain video. The operations are described in detail as follows:

Peer joining. Create a new peer line for this peer, add it to the hash table. The index is initially set to 0, and all the pointers point to NULL. The time complexity is $O(1)$;

Peer leaving. First find the line of that peer, for each non-NULL pointer, update the previous one, making it pointing to the next, and then remove the peer line from the hash table. The time complexity is $O(kN(1 - \mathrm{e}^{-kn/m}))$;

Peer adding new video. First find the line of that peer, map the video ID to k positions. For each position, insert to the first after the head line and update its own pointer. The time complexity is $O(k)$;

Peer delete video. In this case, the peer notifies server or neighbors with the updated index. When receiving the new index, for each position, if the original one is 1 and the new one is 0, then remove the pointer of that position like the operation of peer leaving; if the original one is 0 and the new one is 1, then insert the pointer to the first after the head line like the operation of peer adding new video; if the original one and the new one are the same, then skip to the next bit. The time complexity is $O(kN(1 - \mathrm{e}^{-kn/m}))$;

Searching for all peers having a certain video. First map the video ID to an integer number, as well as to k position numbers, and select the first position number (selecting any of the positions will work). From the head line, find the peer whose index bit of this position is 1, and do a bitwise-AND operation with the mapped integer and the index of this peer. If the result is unchanged, which means this peer is storing this video (with false positive), add this peer to a peer list. Finally return the peer list. The time complexity is $O(N(1 - \mathrm{e}^{-kn/m}))$.

Note that the operation time of some operations depends on the number of online peers, which seems not scalable. However, for peer leaving and peer delete video operations, the bottleneck is network and these two operations have nothing to do with the end user. For the last one, the server only check a small portion of all peers (i.e., 5.7% for $k = 3, n = 5, m = 256$), and since neighbors of each users are not so many (i.e., less than 20 on average, refer to the next chapter), the response time is not inefficient.

4.5 Source Rate Allocation Mechanism

In Section 3.3.4, we know that the bitrate of YouTube video is mostly around 330 kbps. We define a minimum streaming rate $r = 384$ kbps, which is recommended by Adobe in [33], as this minimum streaming rate ensures the continuous playback. We denote a peer's maximum downloading bandwidth as b_d, and uploading bandwidth as b_u, a peer can download from at most *max_down* peers, and can upload to at most *max_up* peers.

If the peer is downloading from the server, the server allocates b_d source rate. Although server provide much more source rate than the minimum requirement, it does not increase the traffic of the server. Moreover, the peer being downloading from the server indicates that there are not enough sources of this video, and thus the sooner the peer can be a source, the sooner the server can reduce a part of the workload. If the peer is downloading from other peers, and the downloading rate is not less than r, the server does not need to provide additional source rate, but if the parent peers cannot provide r source rate totally, the server then provides additional $(b_d - r)$ source rate as another source.

Algorithm 1 ReqDownload(*peer_list*, *video_id*)

```
for each peer $p$ in $peer\_list$ do
  $rate_{promise}[p] \leftarrow p.\mathsf{GetRatePromise}(video\_id)$
end for
if size of $peer\_list < max\_down$ then
  select all peers whose $rate_{promise} > 0$
else
  select top $max\_down$ peers whose $rate_{promise} > 0$
end if
$rate_{all} \leftarrow$ sum of all selected $rate_{promise}$
if $rate_{all} > b_d$ then
  $a \leftarrow b_d / rate_{all}$
else
  $a \leftarrow 1.0$
end if
for each selected peer $p$ do
  $p.\mathsf{OnUpload}(video\_id, a)$
end for
if downloading rate $rate_{down} < r$ then
  request server for remaining source rate with $r - rate_{down}$
end if
```

The Algorithm 1 shows the process of a peer requesting downloading. When a peer

Algorithm 2 GetRatePromise(*video_id*)

```
if peer does not cache $video\_id$ or peer's number of downloading peer $up\_to \geq max\_up$
then
  return 0
else
  return $b_u/(up\_to + 1)$
end if
```

P_a requests downloading video from a source peer P_b, P_b first responses with a promised source rate, which means if P_a becomes a downloading peer, P_b will guarantee a source rate required by P_a, and the required source rate is at least this promised source rate if P_a has enough downloading bandwidth. After P_a collects all the sources' promised source rates, it selects the top *max_down* peers and adds up all their promised source rate $rate_{all}$, and calculates a factor $a = b_d/rate_{all}$. If $a < 1.0$, the sources can provide more source rate than P_a's maximum downloading bandwidth; if $a \geq 1.0$, a is set to 1. Then P_a sends the downloading requests to those selected peers with factor a and its index. The source peers then send back their indexes and begin to send the video file to P_a, and then P_a and those peers become neighbors.

When a new peer becomes the downloading child, the source peer sometimes need to re-allocate the source rate of the existing downloading peers. The main purpose is to balance the source rate allocation, that is, if some peers have much more source rate, the peer will reduce the source rate of those peers. A detailed algorithm is provided as Algorithm 3.

4.6 Pre-Fetching Strategy

In general, the video file has been downloaded before the playback ends, so that the peer has a period of free time[2]. If the system predicts and downloads the next video using the free time, once the user does watch this video, the startup delay can be greatly reduced since the beginning part has already been downloaded, thus improving the playback quality. Note that the prediction is made by the system client instead of server or the human user. This pre-fetching strategy can benefit from the social network, because social network clusters the similar videos, as one is likely to be accessed after another.

[2]A test for 149 video shows that in different network condition, it finishes downloading when playback position is 59% of the video length.

Algorithm 3 OnUpload(*video_id*, *a*)

```
$rate_{grant} \leftarrow a * (b_u/(up\_to + 1))$
**if** $rate_{grant} + rate_{up} \le b_u$ **then**
  $rate_{up} \leftarrow rate_{up} + rate_{grant}$
**else**
  $rate_{slot} \leftarrow (b_u - rate_{grant})/up\_to$
  find all $k$ children whose source rate $rate_{down} > rate_{slot}$
  $rate_{all} \leftarrow$ sum of source rate of those $k$ children
  **for** each $k$ children $p$ **do**
    decrease the source rate to $rate_{all}/k$
  **end for**
  $rate_{up} \leftarrow b_u$
**end if**
$up\_to \leftarrow up\_to + 1$
begin uploading with rate $rate_{grant}$
```

To avoid wasting bandwidth, instead of pre-fetching all the predicted video file, it is better to download the beginning (i.e., a 5-second clip of the beginning) of the predicted video only. Otherwise, once the prediction is incorrect, and if the user will never watch the predicted video, it is a waste of bandwidth.

Since we expect to reduce the workload of the server, the pre-fetching only occurs among peers, that is, even if there is no peer caching the video or the downloading rate is less than the minimum rate, the server will not provide additional source rate. Furthermore, we use a one-hop searching. If the predicted video is not cached by any neighbor peers, the peer will make another prediction. Therefore in pre-fetching, the peer does not send request to its neighbor, because the indexes of its neighbors are store locally.

Suppose after watching a video, every video, including those not in the related video list, has a probability to be selected. This probability can be easily calculated by recording the transactions of users' selections. The system thus can utilize the probabilities to predict. To increase the prediction hit rate, the system can predict multiple videos, so that the peer can download several beginning clips. The prediction accuracy will be analyzed next.

Since we do not have the trace of users' selections, we have to make some assumptions, as the simulation in the next chapter is also based on these assumptions. We assume a user only select the next video in the related video list. Suppose there are r related videos in order, the user will select the video randomly with a probability that is inversely proportional to the order of that video, that is, to select ith video with the probability of $\frac{1}{i \cdot s}$, where

$s = \sum_{i=1}^{r}(\frac{1}{i})$. The system also predicts the next video with this probability. Considering predicting once, the probability of actually selecting video i is

$$\mathrm{P}(X = i) = \frac{1}{i \cdot s},$$

and the probability of predicting video i is

$$\mathrm{P}(Y = i) = \frac{1}{i \cdot s},$$

so that the probability of correct prediction is

$$\begin{aligned} & \sum_i \mathrm{P}(Y = X) \\ = & \sum_i \mathrm{P}(Y = i, X = i) \\ = & \sum_i \mathrm{P}(Y = i) \cdot \mathrm{P}(X = i). \end{aligned}$$

Considering predicting k times, the probability of correct prediction is

$$\begin{aligned} & 1 - \prod_k \Big(\mathrm{P}(Y_k \neq X_k)\Big) \\ = & 1 - \prod_k \Big(\sum_i \mathrm{P}(Y_k = i, X_k \neq i)\Big) \\ = & 1 - \prod_k \Big(\sum_i \mathrm{P}(Y_k = i) \cdot \mathrm{P}(X_k \neq i)\Big) \\ = & 1 - \prod_k \Big(\sum_i \mathrm{P}(Y_k = i) \cdot (1 - \mathrm{P}(X_k = i))\Big). \end{aligned}$$

We plot the estimated prediction accuracy as a function of number of related videos for different times of predictions in Figure 4.4. Not surprisingly, from the figure we can see that when the number of related videos increases, the prediction accuracy decreases, and the more it predicts, the more accurate the prediction is. Suppose the average number of related videos is 10, predicting 5 times can obtain a 65% accuracy approximately, and downloading 5 video clips transfers about 1 MB data (suppose each clip is about 200 KB), which is quite acceptable.

4.7 Summary

In this chapter, we first summarize the challenges and potentials of utilizing peer-to-peer in short video sharing system. We present a novel social network based short video sharing

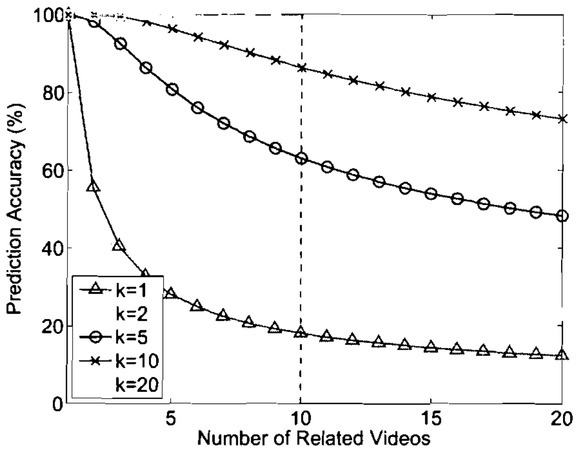


Figure 4.4: Prediction accuracy to number of related videos for different times of predictions

design. To efficiently utilize peer-to-peer in this system, we proposed solutions addressing some key design issues, including a novel bi-layer overlay that explores the social network, an indexing scheme to store and search the information of peers' cached videos efficiently, a source rate allocation mechanism that guarantees the continuous playback, and a pre-fetching strategy that decreases the startup delay of establishing P2P overlay. We have performed extensive simulations and experiment to evaluate our system. In the next chapter, we will present the simulation results that demonstrate the good performance of our system.

Chapter 5

System Evaluation

We have performed extensive simulations and experiments. In this chapter, we first illustrate the test dataset of the simulation, and introduce the simulation configurations. Then we systematically present the experiment results, which demonstrate that our system greatly reduce the workload of server and improve the quality of playback. We also investigate the effectiveness of social network in our system.

5.1 Methodology of Simulation

5.1.1 The Dataset Description

On July 14th, 2007, we crawled more than 100 thousand YouTube videos information with our YouTube crawler. The crawled data do not depend on the previous crawled data, so there is overlap with the data for the measurement study.

We have done pre-processing for the data as follows: we only use the information of video ID, length, views and related videos; we remove the videos that are shorter than 5 seconds or longer than 720 seconds; we remove the related video ID that does not appear in this database, and remove the videos that do not have related videos (the related videos might be removed); we do this iteratively until there is no change.

The result database contains 96,709 distinct videos, each contains an ID, a length, a number of views and a list of related videos that contains at least one and at most 20. The average out-degree is 9.52, and the maximum in degree is 165. The videos graph forms a small-world: the clustering coefficient is 0.529, and the characteristic path length of its LCC

(Largest Connected Component) is 14.1.

5.1.2 Configuration and Simulation Settings

Index Configurations

For indexing scheme, we set m as 256 and k as 3. For searching all peers caching a certain video, the operation return at most 32 peers' addresses. We modify the sdbm hash function [65] to k independent hash functions for our index scheme. The algorithm is as follows:

Algorithm 4 HashFunction($video\_id$, k)

```
$hash \leftarrow 0$
for $i = 1$ to 11 do
    $hash \leftarrow video\_id[i] + (hash << (6 - k)) + (hash << (16 - k * 2)) - hash$
end for
$tmp \leftarrow (\mathbf{int})(hash/10000)$
$hash \leftarrow ((hash - tmp) * 10000 + tmp)$ mod 256
return $hash$
```

New Peer Arrival Rate

We assume new peer joining the system following a Poisson distribution with $\lambda = 6$, which results in an approximate total number of $6 \times 86400 \times 30 = 1.55 \times 10^6$ users access the system per month (YouTube has 2×10^7 visits per month [66]). The total simulation time is 6 hours.

Bandwidth Capacity and Video Bitrate

The study in [45] shows the breakdown of the bandwidth capacity of current Internet users. We modify the statistic slightly and list as Table 5.1. We randomly assign a peer's bandwidth capacity with a probability according to the distributions in the table.

	Low	**DSL1**	**DSL2**	**Cable and Ethernet**
Downloading	256 kbps	768 kbps	1500 kbps	3000 kbps
Uploading	256 kbps	128 kbps	384 kbps	768 kbps
Distribution	7.1%	14.3%	23.3%	55.3%

Table 5.1: Bandwidth capacity and distribution of users

The video bitrate is set to 330 kbps according to Section 3.3.4. Therefore for the first type of peers, called *low bandwidth peers*, the playback delay is inevitable, and we call the others as *high bandwidth peers*. The minimum streaming rate requirement is 384 kbps, which is recommended in [33].

Selecting, Watching and Caching Video

The user selects the first video with the probability that is proportional to the number of views of the videos. This results in a small number of videos have been watched many times but a great number of videos are seldom watched, and causes the scale-free characteristic in the video views.

The user selects the next video according to the illustion in Section 4.6: the user only select the next video in the related video list with a probability that is inversely proportional to the order of that video. Therefore, the video in the top list is likely to be selected. Moreover, the user does not select the video that is already watched. The user begins to select next video after 5 second when finished watching the previous one, and note that this waiting time is not the delay we discuss next.

Each peer will watch the entire video. If a peer cannot select another video (i.e., all the related videos have been watched), it will leave the system. Low bandwidth peers will leave the system after watching 5 videos, and others will leave after watching 10 videos.

Each peer can cache at most 5 videos. When the cache is full, the peer will drop the earliest cached video.

Downloading, Uploading and Pre-fetching Video

The downloading and uploading source rate is allocated according to Section 4.5: when downloading only from the server, the peer utilizes the maximum downloading bandwidth; when downloading from other peers, the peer just need to satisfy the minimum streaming rate requirement. A peer can download from at most 10 other peers, and can upload to at most 5 other peers.

In simulation with pre-fetching, the peer will pre-fetch a 5-second video clip after finish downloading the current one. The peer only pre-fetch video from its neighbors. If none of its neighbors have cached the predicted video, it will predict another one in the next time unit. When pre-fetching the video, even if the neighbors cannot provide enough source rate, the server will not provide the additional source rate. If the peer have not finished pre-fetching a video clip when the user finishes watching the current one, the pre-fetching will stop and currently pre-fetching video clip will be discarded. The maximum number of pre-fetched video clips is 20.

Delay

There are three types of delays in the simulation: startup delay, downloading delay and playback delay.

The startup delay is the delay before start downloading a video. After selecting a video, the peer first need to download a page that embeds the video and other information, as we define the page size as 80 KB, which is the typical YouTube webpage size. Then before start downloading a video, the peer requests peer-list and request downloading. The startup delay consists of the transmission delay calculated by the request size and its bandwidth and the propagation delay as 10 ms, which is a typical round-trip-time. We ignore the processing delay and queueing delay. If a peer has pre-fetched the video that it is going to download, the startup delay may be minimized to 0. Since our simulation is time-driven, the minimum time unit is one second. If the delay is greater than 0, we calculate the ceiling of the integer number of the delay. However, we output the exact float number of the delay.

The peers periodically (every 30 seconds) check the overlay, exploring more sources for more source rate. Since there are requests transmitted, the downloading will be disrupted, as we call the delay as downloading delay. The downloading delay generally does not affect the playback of the user, but stops downloading for a while.

The third delay is playback delay. When encountering playback delay, which means the current playback position has not been downloaded during the playback, the peer will stop watching for 3 seconds. At the meantime, the downloading continues, and then the peer goes on watching.

Baseline for Comparison

We have also performed two baseline simulations. The goal of the first simulation is to compare our peer-to-peer system with traditional peer-to-peer system, in which the peers do not cache videos that they have downloaded, and can only upload the currently watching video to those peers whose playback positions are behind theirs.

For the second baseline simulation, we expect to understand the effectiveness of social network in our system. Therefore, this simulation does not have the social network, as videos do not have related videos, so that they are independent with each other. To perform this simulation, we first get the distribution of the count of watched videos and the count of pre-fetched videos from the normal simulation. In this baseline simulation, peers select every video as same as selecting the first one. And whether or not the peer leaves the system is according to the distribution of the previous simulation.

5.2 Simulation Results

We present the simulation results in this section. We first show the result of the system performance in terms of the server and the end user, in particular, the server bandwidth consumption and quality of playback, and then we show the peer behavior in the simulation. The effectiveness of social network in our system is also studied. At last, we examine the performance of the indexing scheme.

5.2.1 Bandwidth Consumption

Our primary goal is to reduce the server bandwidth consumption. This can be done by implementing peer-to-peer in the system, since peers will contribute their uploading bandwidth. Figure 5.1 compares the bandwidth consumption in C/S architecture, traditional P2P architecture and social network based P2P architecture.

The system starts from no peer being in the system, and thus all the three increase as the number of online peers increases. After one hour, the system is almost stable. We can see that the bandwidth consumption in traditional P2P is about 56.3% of that in C/S architecture, and the bandwidth consumption in our design is about 32.2% of that in C/S architecture. Therefore, peers in our system contribute nearly 70% total bandwidth.

We also find that among peers' total bandwidth contributed, about 14.0% is for pre-fetching. Note that server is not involved in pre-fetching, considering the bandwidth consumption of pre-fetching is quite low, and the impact on startup delay we will study next, we can see the importance of pre-fetching.

The main difference between the traditional P2P architecture and social network based P2P architecture is that in traditional P2P, peers only upload the video that they are watching, whereas in our design, peers can upload all the videos they have cached. Therefore,

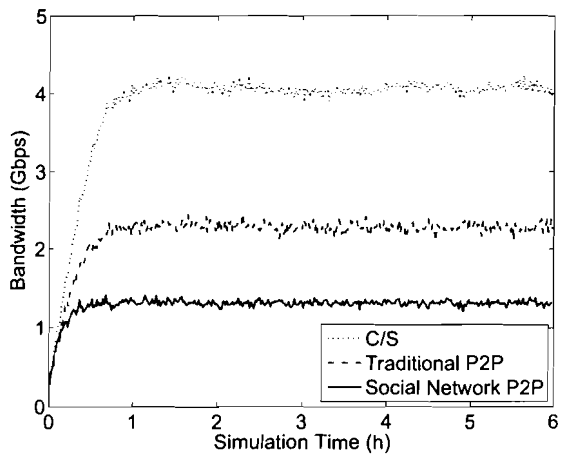


Figure 5.1: Comparison of bandwidth consumption

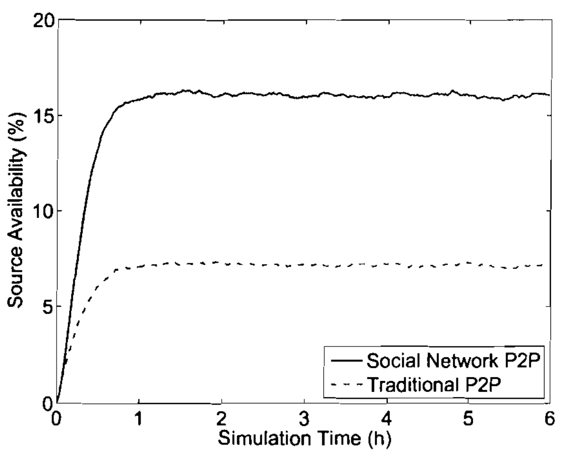


Figure 5.2: Comparison of video source availability

the sources in the entire system are much more in our design, as shown in Figure 5.2. In traditional P2P, about 7.1% videos have been downloaded and at least one peer can be the source, but in our design, the number is 16.1%, and thus more videos can be uploaded by the peers, so that it greatly increases the scalability of the system.

5.2.2 Quality of Playback

Playback quality is important to the end user. In particular, we expect a short startup delay and continuous playback in our system.

Startup Delay Distribution

In our system, startup delay mainly consists of the time to download the page, the time to search for peers and the time to request downloading from peers. If the peer has already pre-fetched the page and beginning of the video, the startup delay can be much shortened. Figure 5.3 compares the CDF of startup delays in simulations with and without pre-fetching.

We can see that with pre-fetching, about 42.9% of the peers do not have startup delay, which means they do not need to wait to start watching the next video. If fail to correctly pre-fetch the next video, the startup delay is at least 200 ms in both simulations. It is still quite short because in our system, the peers only search for peers within two hops. This also shows that the design of bi-layer overlay is good. We find that there are four curves on the entire CDF line, because of the four types of different bandwidth capacity.

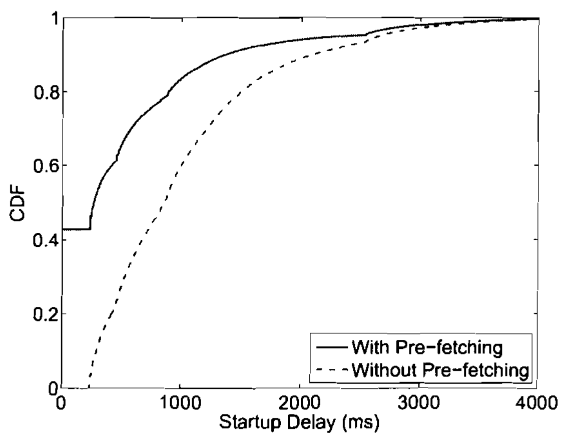


Figure 5.3: CDF of startup delay

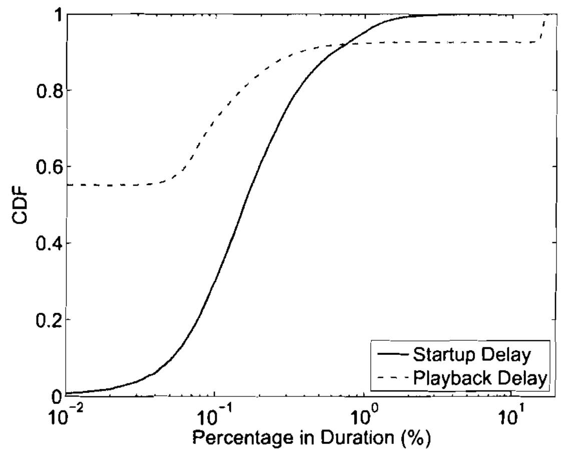


Figure 5.4: CDF of percentage of delay in duration

Playback Delay Distribution

Playback delay is due to the low downloading source rate (lower than the video's bitrate). In Section 5.1.2, we know that only the first type of peers' maximum downloading bandwidth is lower than the video's bitrate, so these peers have to wait for some time and then continue to watch. For high bandwidth peers, as long as their downloading source rate is guaranteed, they will not encounter playback delay.

However, those high bandwidth peers will possibly encounter playback delay in our system, because their parent peers may leave the system or drop the cached video, thus leaving the overlay. In our system, the peers will periodically check their downloading source rate and explore for more sources to assure the streaming rate. In the simulation, we find that 0.9% peers encounters playback delay at a certain time, and only 5.5% among them are high bandwidth peers.

We plot the percentage of playback delay as well as the startup delay in duration of each peer in Figure 5.4. Note that the x-axis is log-scale. We can see that only less than 10% peers have the total playback delay larger than 0.3% of their duration, as these are probably the low bandwidth peers. Therefore, we can claim that most peers have a continuous playback.

5.2.3 Peer Behavior

In our simulation, we find that the number of online peers grows linearly from zero to about 12 thousand in one hour. Then the peer joining rate and peer leaving rate are balanced, and thus the system is stable. In this thesis, we do not consider massive joining and leaving, as we list it as a future work in Chapter 5.

Percentage of Downloading, Uploading and Pre-fetching Peers

Figure 5.5 plots the percentage of peers that are downloading, uploading and pre-fetching. Note that pre-fetching peers are not included in downloading peers. We can see that only 59.5% peers are downloading videos, if we do not consider pre-fetching. This is because most peers have a downloading bandwidth much higher than the video bitrate, and thus most peers are idle after finish downloading the video. This also enlightens us the idea of pre-fetching. In the simulation, 19.7% peers are pre-fetching, thus further utilizing the downloading bandwidth. We can also see that 65.3% peers are uploading videos, indicating that most of peers are contributing their uploading bandwidth.

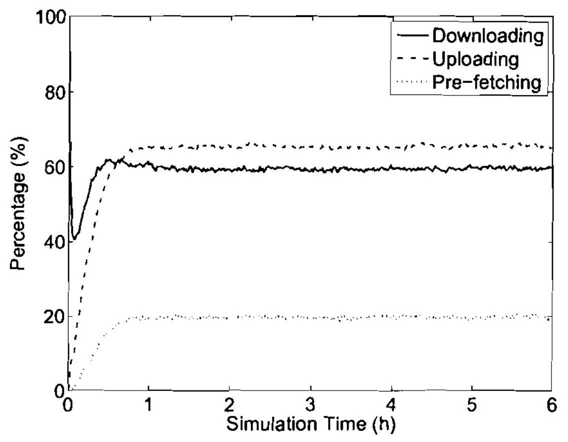


Figure 5.5: Percentage of downloading, uploading and pre-fetching peers

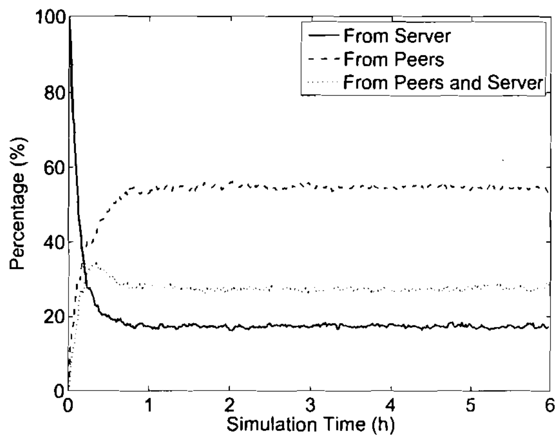


Figure 5.6: Percentage of different downloading source

Percentage of Different Downloading Sources

Next we examine the downloading source in the simulation. In particular, peers can download video only from server, or only from other peers, or from server and peers. Figure 5.6

plots this percentage. In the beginning, most peers are downloading videos from server, because most videos have not been downloaded, and thus there are few sources. After the system is stable, a great number of videos have been cached by peers, so that the peers become the main sources, and the server acts as a backup source. As a result, average 54.7% peers are downloading only from other peers, average 17.3% peers are downloading only from server, and average 27.4% peers are downloading from both server and peers.

Duration Distribution and Modeling

Next we study the duration of peers in the simulation. Figure 5.7 plots the CDF of peers' durations. The curve can be well fitted by a normal distribution, which is also shown in the figure. Therefore, we know that the average duration is 1938 seconds.

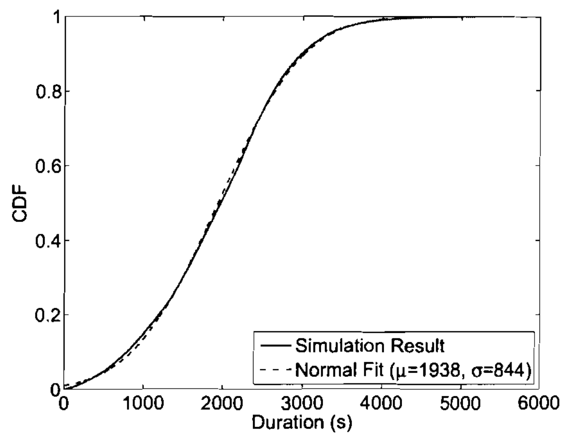


Figure 5.7: CDF of peer duration

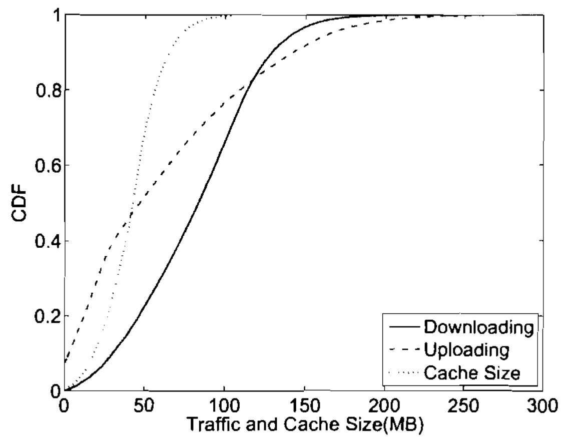


Figure 5.8: CDF of downloading/uploading traffic and cache size

Downloading/Uploading Traffic and Cache Size

At last we study the downloading/uploading traffic and cache size of each peer. Figure 5.8 plots the CDF of the three statistics. The average of the downloading and uploading traffic are 83.2 MB and 62.5 MB, respectively. Therefore we can see peers contribute about three-fourths as they obtain, as this is quite a large proportion. Also we can see that the average cache size is 43.5 MB for each peer, which is quite acceptable (the average number of cached videos is 4.8).

5.2.4 Effectiveness of Social Network

As we emphasized, our system explores the social network, we thus study the effectiveness of social network in our design.

Number of Neighbors and Searched Sources

We first look at the number of neighbors of a peer in the simulations with and without social network. As Figure 5.9 plots the number of neighbors during the simulation, we find that peers have average of 18 and 30 neighbors in two simulations, respectively. The number is greater in the simulation without social network, because neighbors are no longer having similar preferences, and that having the same next video are fewer, so the peer has to search for more sources. However, in the simulation with social network, it is more possible that the peer's current neighbors have the next video, and thus the peer does not need to explore for more. Having less neighbors indicates that less peers will be contacted, thus reducing the communication overhead.

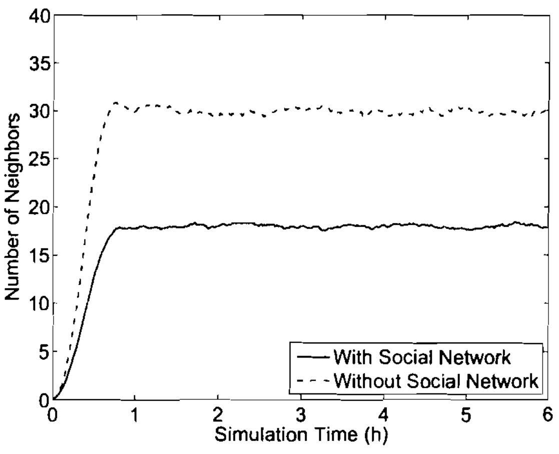


Figure 5.9: Number of neighbors

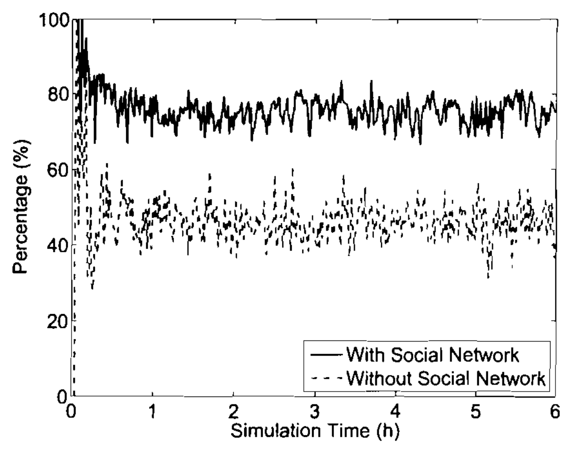


Figure 5.10: Percentage of searched peers

We record the number of source peers in the searched peer-list every time a peer request to its neighbors, as well as the number after request to server, so that we can calculate the percentage of sources returned from neighbors, and thus learn that how many peers that cached the expected video can be found within two hops. Figure 5.10 compares the percentages in the simulations with and without social network. We can see that 75.1% peers that have cached the target videos can be found within two hops in simulation with social network, and only 46.7% peers can be found if without social network. Consider the number of neighbors contacted, the difference will be larger. Therefore, we learn that social network has great impact on our bi-layer overlay design, as three-fourths of the sources can be found within two hops based on social network. In summary, our bi-layer overlay design takes advantage of social network.

Impact on Pre-fetching

In our simulation, we record the number of pre-fetched video clips when the peers are going to select the next one. We find that peers pre-fetch average of 4.1 video clips, and 28.1% of times a peer fails to pre-fetch a single video clip before the next one. This high ratio is because peers only pre-fetch from their neighbors, so if none of the neighbors has the predicted video, the pre-fetch is failed. We plot the probability that a peer's neighbors have the predicted video in Figure 5.11. We can see that the probability in the simulation with social network (average 0.186) is much higher than that in the simulation without social network (average 0.032), suggesting that with social network, neighbors are more likely to have the predicted video.

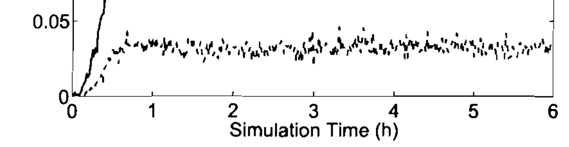


Figure 5.11: Probability of having the predicted video

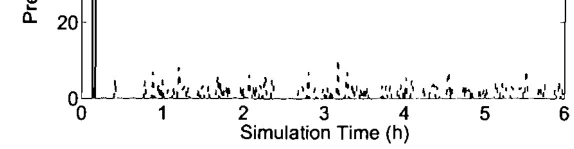


Figure 5.12: Accuracy of pre-fetching

Figure 5.12 compares the accuracy of pre-fetching in the simulations with and without social network. The accuracy in the simulation with social network is about 59.6%, which is quite close to our modeling in Section 4.6, and is much higher than that in the simulation without social network (about 1%), because without social network, the system has to predict the next video from the entire video repository.

5.2.5 Effectiveness of Indexing

We finally study the effectiveness of our indexing scheme.

Key Distribution

We first examine the performance of the hash functions in a separate experiment. In particular, we exam the number of same hashed key of different video IDs among our dataset. From Table 5.2, we can see that the keys of the hashed video IDs are well distributed, suggesting that our hash functions are effective and can well control the probability of false positive in Bloom filter.

Appearance	Count	Percentage
once	93545	96.73%
twice	3116	3.22%
trice	48	0.05%

Table 5.2: Distribution of hashed video ID keys

False Positive

Next we examine the false positive probability during the simulation. The false positive in our system increases the overhead, and thus increases the startup delay.

The theoretical false positive probability is not suitable for this case, because it does not consider the different combination of different position with bit "1". Therefore we propose the false positive probability in our case is $f = C_k^{kn} \cdot (1 - e^{-kn/m})^k$, so that we calculate the probability is $f = C_3^{3\times5} \cdot (1 - e^{-3\times5/256})^3 = 0.084$. However, we find that the result probability is higher than that, about 0.183. This is because videos are not selected with the equal probability, as there exists high temporal locality of request, which means most of the videos have not been requested but a few videos have been requested many times. If a request of a popular video turns out to be false positive, the probability is increased greatly. Nevertheless, the false positive probability is still acceptable.

Indexing Table Size

At last we look at the server's indexing table size. We find that the size is dependent on the number of online peers. On average, it requires less than 1 KB for each peer. For total 12 thousand online peers, the server requires 12 MB for the indexing table. This small size is quite acceptable for the server.

5.3 Summary

We have performed extensive simulations and experiments, and present the result in this chapter. The results demonstrate that our design greatly increases the number of video sources, and thus reduce the server workload significantly, about nearly 70%. Using pre-fetching results in nearly half of the peers do not have startup delay. Our rate allocation mechanism assure most of high bandwidth peers will not encounter playback delay. We also study the behaviors of peers in the system, finding that most peers contribute about three-fourths as they obtain. The social network not only increases the accuracy of pre-fetching, but also makes the search for sources more efficient.

Chapter 6

Conclusion and Future Work

In this chapter, we first conclude this thesis, and then note some possible future directions for extending it.

6.1 Conclusion

In this thesis, we first present a detailed measurement study on the characteristics of YouTube videos. Through examining more than 3 million video data collected in a four-month period, we have demonstrated that, while sharing certain similar features with traditional video repositories, YouTube exhibits many unique characteristics, especially in video length distribution. These characteristics introduce novel challenges and opportunities for optimizing the performance of short video sharing services.

In the measurement study, we have also investigated the social networking aspect in YouTube videos, which is probably the most unique and interesting aspect, and has substantially contributed to the success of this new generation of service. We have found that the networks of related videos, which are chosen based on user-generated content, have both small-world characteristics of a large clustering coefficient indicating the grouping of videos, and a short characteristic path length linking any two videos. We suggest that these features can be exploited to facilitate the design of novel caching and peer-to-peer strategies for short video sharing.

Next we summarize the challenges and potentials of utilizing peer-to-peer in short video sharing system and present a novel social network based short video sharing design. To efficiently utilize peer-to-peer in this system, we address a series of key design issues, and present novel solutions including a bi-layer overlay that leverages social network, an indexing scheme to store and search the information of peers' cached videos efficiently, a source rate allocation mechanism to guarantee the continuous playback, and a pre-fetching strategy to decrease the startup delay of establishing peer-to-peer overlay.

We have performed extensive simulations and experiments to evaluate our system design. The results demonstrate that our system can reduce nearly 70% of the server workload, and peers contribute three-fourths as they obtain. Pre-fetching strategy results in nearly half of the peers do not have startup delay. More importantly, the social network not only increases the accuracy of pre-fetching, but also makes the search for peer more efficient. These results show that our design successfully takes advantage of peer-to-peer technique and property of social network.

6.2 Future Work

The first thing we expect to work on is to implement a prototype on *PlanetLab* [16]. PlanetLab is a global research network that supports the development of new network services, and currently consists of 842 nodes at 416 sites. PlanetLab has been used to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. Experiment on PlanetLab can better reflect the networking characteristics than the simulation, because it is real-world experiment. This work, however, is challenge, because in our design, the server still exists and consumes a great amount of bandwidth comparing to a single peer, but there is bandwidth limited for each node in PlanetLab. This problem can be addressed by scaling down the bandwidth, file size and bitrate. Another challenge is that, there are less than one thousand nodes in PlanetLab, whereas our design consists of more than ten thousand simultaneous peers. A effective approach should be proposed to tackle this problem.

Another extended work is the searching for sources. In our design, we use a two-hop flooding for searching, and thanks to social networks, this two-hop flooding can return about 75% sources. If searching for several more hops, all the expected peers are likely to be found, thus offloading the server further. However, since our overlay is mesh based, flooding will cause a great amount of redundancy, hence increasing the overhead. Therefore, we expect to establish an efficient approach to search for more sources.

Since we do not obtain the trace of each user's selections, we have to make several assumptions on how the user selects the next video and how the system predicts the next video. In a real system, if the server can store these transactions, it can perform real-time data mining on the frequent pattern, and calculates a probability for each pair of videos that are consecutively selected frequently, so that it can benefit the pre-fetching greatly.

In this thesis, we do not consider massive joining and leaving of the peers, as in our system, the peer joining the system follows a Poisson distribution, and the result duration follows a normal distribution. When suddenly a great number of peers joining or leaving the system (referred to as flash crowd), the system performance is inevitably affected. This feature is worth investigating.

The next extended direction is the piece-selection strategy. Some of the related works study this problem, yet we do not focus on it in this thesis. Since short video media is different from traditional media, the previous techniques need to be revised to be implemented in short video sharing system, i.e., the strategy of how to determine the piece size and how to scheduling the transmission.

Finally, implementing network coding in this system is promising. Network coding [21] is found that can not only reduce the traffic but also increase the robustness, as peers can obtain the original data when receive any required number of encoded data. However, peers require more time to decode the video file. This direction is also worth studying.

Bibliography

[1] Alexa. http://www.alexa.com.

[2] API Documentation (YouTube). http://youtube.com/dev_docs.

[3] BitTorrent. http://www.bittorrent.com.

[4] Blog (YouTube). http://youtube.com/blog.

[5] CoolStreaming. http://www.coolstreaming.us.

[6] Daum UCC. http://ucc.daum.net.

[7] Facebook. http://www.facebook.com.

[8] Flickr. http://www.flickr.com.

[9] GridCast. http://www.gridcast.cn.

[10] Joost. http://www.joost.com.

[11] LiveJournal. http://www.livejournal.com.

[12] Windows Live Spaces. http://spaces.live.com.

[13] MSN Video. http://video.msn.com.

[14] MySpace. http://www.myspace.com.

[15] Orkut. http://www.orkut.com/Home.aspx.

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

[17] PPLive. http://www.pplive.com.

[18] Tudou. http://www.tudou.com.

[19] YouTube. http://www.youtube.com.

[20] Soam Acharya, Brian Smith, and Peter Parnes. Characterizing User Access To Videos On The World Wide Web. In *Proceeding of the 8th ACM/SPIE Multimedia Computing and Networking (MMCN'00)*, 2000.

[21] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[22] Réka Albert, Hawoong Jeong, and Albert-Lászlő Barabási. The Diameter of the World Wide Web. *Nature*, 1999.

[23] Jussara Almeida, Jeffrey Krueger, Derek Eager, and Mary Vernon. Analysis of Educational Media Server Workloads. In *Proceeding of the 11th ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, 2001.

[24] David Best. Web 2.0: Next Big Thing or Next Big Internet Bubble. Technical report, Technische Universiteit Eindhoven, 2006.

[25] Burton Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.

[26] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceeding of the 18th IEEE Conference on Computer Communications (INFOCOM'99)*, 1999.

[27] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485–509, 2003.

[28] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proceeding of ACM Internet Measurement Conference (IMC'07)*, 2007.

[29] Bin Cheng, Xuezheng Liu, Zheng Zhang, and Hai Jin. A Measurement Study of a Peer-to-Peer Video-on-Demand System. In *Proceeding of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07)*, 2007.

[30] Ludmial Cherkasova and Minaxi Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. In *Proceeding of the 12th ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02)*, 2002.

[31] Maureen Chesire, Alec Wolman, Geoffrey Voelker, and Henry Levy. Measurement and Analysis of a Streaming-Media Workload. In *Proceeding of the 3rd conference on USENIX Symposium on Internet Technologies and Systems (USITS'01)*, 2001.

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

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

[34] Yi Cui, Baochun Li, and Klara Nahrstedt. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. *IEEE Journal on Selected Areas in Communications*, 22(1):91–106, 2004.

[35] Tai Do, Kien Hua, and Mounir Tantaoui. P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment. In *Proceeding of the IEEE International Conference on Communications (ICC'04)*, 2004.

[36] Peter Sheridan Dodds, Roby Muhamad, and Duncan Watts. An Experimental Study of Search in Global Social Networks. *Science*, 301(5634):827–829, August 2003.

[37] Ellacoya Data Shows Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network.
http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf, 2007.

[38] Pawel Garbacki, Dick Epema, Johan Pouwelse, and Maarten van Steen. Offloading Servers with Collaborative Video on Demand. In *Proceeding of the 7th International Workshop on Peer-to-Peer Systems (IPTPS'08)*, 2008.

[39] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube Traffic Characterization: A View From the Edge. In *Proceeding of ACM Internet Measurement Conference (IMC'07)*, 2007.

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

[41] Larry Greenemeier and Sharon Gaudin. Amid The Rush To Web 2.0, Some Words Of Warning. `http://www.informationweek.com/news/showArticle.jhtml?articleID=199702353`, 2007.

[42] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2Cast: Peer-to-Peer Patching Scheme for VoD Service. In *Proceeding of the 12th ACM International Conference on World Wide Web (WWW'03)*, 2003.

[43] Martin Halvey and Mark Keane. Exploring Social Dynamics in Online Media Sharing. In *Proceeding of the 16th ACM International Conference on World Wide Web (WWW'07) Poster Paper*, 2007.

[44] Theodore Hong. Performance. In Andy Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 14, pages 203–241. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[45] Cheng Huang, Jin Li, and Keith Ross. Can Internet Video-on-Demand be Profitable? In *Proceeding of ACM SIGCOMM'07*, 2007.

[46] Cheng Huang, Jin Li, and Keith Ross. Peer-Assisted VoD: Making Internet Vieo Distribution Cheap. In *Proceeding of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07)*, 2007.

[47] Xiaofei Liao and Hai Jin. OCTOPUS: A Hybrid Scheduling Strategy for P2P VoD Services. In *Proceeding of the 6th IEEE International Conference on Grid and Cooperative Computing (GCC'07)*, 2007.

[48] Gang Liu, Mingzeng Hu, Binxing Fang, and Hongli Zhang. Measurement and Modeling of Large-Scale Peer-to-Peer Storage System. In *Proceeding of Grid and Cooperative Computing Workshops (GCC'04)*, 2004.

[49] Jiangchuan Liu and Jianliang Xu. Proxy Caching for Media Streaming over the Internet. *IEEE Communications Magazine*, 42(8):88–94, 2004.

[50] Stanley Milgram. The Small World Problem. *Psychology Today*, 2(1):60–67, 1967.

[51] Alan Mislove, Massimiliano Marcon, Krishna Gummadi, Peter Dreschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceeding of ACM Internet Measurement Conference (IMC'07)*, 2007.

[52] Jan David Mol, Johan Pouwelse, Michel Meulpolder, Dick Epema, and Henk Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems. In *Proceeding of the 15th SPIE/ACM Multimedia Computing and Networking (MMCN'08)*, 2008.

[53] Tim 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.

[54] John Paolillo. Structure and Network in the YouTube Core. In *Proceeding of the 41st Hawaii International Conference on System Sciences (HICSS'08)*, 2008.

[55] Erzsébet Ravasz and Albert-László Barabási. Hierarchical Organization in Complex Networks. *Physical Review E*, 67(2):026112, 2003.

[56] Subhabrata Sen, Jennifer Rexford, and Donald Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceeding of the 18th IEEE Conference on Computer Communications (INFOCOM'99)*, 1999.

[57] Social network (Wikipedia). `http://en.wikipedia.org/wiki/Social_network`.

[58] Social network service (Wikipedia).
`http://en.wikipedia.org/wiki/Social_networking_service`.

[59] Wenting Tang, Yun Fu, Ludmila Cherkasova, and Amin Vahdat. Long-term Streaming Media Server Workload Analysis and Modeling. Technical report, HP Labs, 2003.

[60] Eveline Veloso, Virgílio Almeida, Jr. Wagner Meira, Azer Bestavros, and Shudong Jin. A Hierarchical Characterization of a Live Streaming Media Workload. *IEEE/ACM Transactions on Networking (TON)*, 14(1):133–146, 2006.

[61] Video on demand (Wikipedia). `http://en.wikipedia.org/wiki/Video_on_demand`.

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

[63] Duncan Watts and Steven Strogatz. Collective Dynamics of "Small-World" Networks. *Nature*, 393(6684):440–442, 1998.

[64] Web 2.0 (Wikipedia). `http://en.wikipedia.org/wiki/Web_2.0`.

[65] Ozan Yigit. Hash Functions. `http://www.cse.yorku.ca/~oz/hash.html`, 2003.

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

[67] YouTube - Video Format (Wikipedia).
`http://en.wikipedia.org/wiki/Youtube#Video_format`.

[68] YouTube video-sharing site is changing popular culture.
`http://www.kcrw.com/news/programs/ww/ww061122youtube_video-sharin`, 2006.

[69] Hongliang Yu, Dongdong Zheng, Ben Zhao, and Weimin Zheng. Understanding User Behavior in Large-Scale Video-on-Demand Systems. *ACM SIGOPS Operating Systems Review*, 40(4):333–344, 2006.

[70] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proceeding of the 24th IEEE Conference on Computer Communications (INFOCOM'05)*, 2005.

[71] Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose. Watch Global, Cache Local: YouTube Network Traffic at a Campus Network - Measurements and Implications. In *Proceeding of the 15th SPIE/ACM Multimedia Computing and Networking (MMCN'08)*, 2008.