Watch Global, Cache Local: YouTube Network Traffic at a Campus Network - Measurements and Implications

Michael Zink, Kyoungwon Suh, Yu Gu and Jim Kurose

Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA

ABSTRACT

User Generated Content has become very popular since the birth of web services such as YouTube allowing the distribution of such user-produced media content in an easy manner. YouTube-like services are different from existing traditional VoD services because the service provider has only limited control over the creation of new content. We analyze how the content distribution in YouTube is realized and then conduct a measurement study of YouTube traffic in a large university campus network. The analysis of the traffic shows that: (1) No strong correlation is observed between global and local popularity; (2) neither time scale nor user population has an impact on the local popularity distribution; (3) video clips of local interest have a high local popularity. Using our measurement data to drive trace-driven simulations, we also demonstrate the implications of alternative distribution infrastructures on the performance of a YouTube-like VoD service. The results of these simulations show that client-based local caching, P2P-based distribution, and proxy caching can reduce network traffic significantly and allow faster access to video clips.

Keywords: Measurement study, Peer-to-peer, Content distribution, Caching

1. INTRODUCTION

During the past two years YouTube has become a very popular web application. Viewers around the world request millions of videos every day*. So far, there is no indication that this popularity will decrease in the near future, and may rather increase, video clips from broadcasters and media companies being streamed to mobile devices.

YouTube represents a service that is different from the traditional VoD systems. On one hand, traditional VoD systems usually offer professionally produced video content like movies, news, sport events or TV series. The quality and popularity of these content can often be controlled and predicted. In contrast, YouTube video contents are uploaded by anyone with access to the network. The content and quality of these video clips vary vastly. As a consequence, predicting how many new videos will be uploaded and their popularity within the YouTube community is very hard. The manner in which content spreads from these two systems to the public is also different. In the VoD system, either the viewers expect regular updates on the content (in the case of news or TV series†) or the content provider can announce new content (e.g., the upcoming of the latest block buster). In the YouTube system, it is often the case that a video clip has become extremely popular after viewers became aware of the clip and told their friends about it, discussed it in BLOGs, and put embedded links to the clip on their own web pages.

These distinct features of YouTube type service lead to many interesting questions. In this work, we focus on how the user request characteristics of YouTube could potentially affect the effectiveness of various content distribution paradigms. Content distribution systems have been thoroughly investigated for traditional VoD systems in the past. For example, for CDNs, it was assumed that the expected popularity of the content in the VoD system is high so that this content should be available from a large portion of content servers in the VoD system. However, in the case of YouTube, the unpredictability of YouTube content popularity invalidates this assumption.

Further author information: E-mail: zink@cs.umass.edu, Telephone: 1 413 545 4465

*`http://news.zdnet.com/2100-9588\_22-6094687.html`

†Many TV channels already make a large portion of their TV series available via their web sites after they have been originally broadcast on traditional TV. In this case, viewers know when a new sequel will be available.

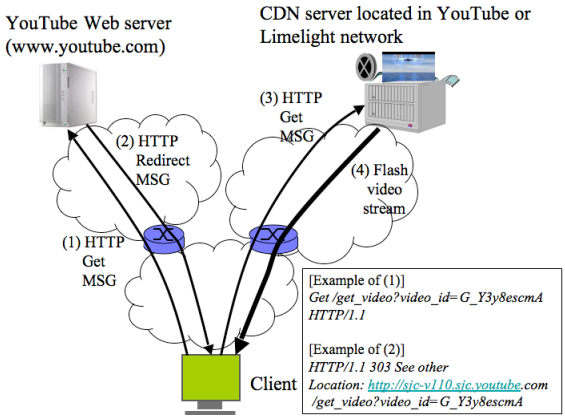

Figure 1. Video retrieval in YouTube

Our work begins with a measurement study of YouTube traffic. The measurement study utilizes network traces that were collected by monitoring traffic at the interconnection between a University campus access network and the Internet. We looked into the time series of user requests for YouTube videos and investigated how video clips are requested by the viewers. We also obtained the global popularity of video clips from the YouTube web site and compared that with the popularity obtained from the local network trace. Our study reveals that local and global popularity information is uncorrelated. Thus, global popularity information from YouTube does not necessarily reflect the popularity of video clips in a local access network and might not be used to support local caching mechanisms. A significant number of video clips are requested twice or more during a short time span and caching in the local access network can reduce overall network load. The result from a trace-driven simulation shows that, in the case of local caching with a cache size of 50 MByte, 84% of request for video clips with multiple requests from the same client can be served from the client's local cache. In the case of proxy caching, the theoretical upper limit for the cache hit rate can almost be reached.

The rest of the paper is structured as follows. Section 2 describes the basic interaction between a YouTube server and a client browser. The monitoring environment and the monitoring procedure of YouTube traffic are described in Section 3 and an analysis of the trace data is presented in Section 4. In Section 5, the impact of alternative distribution infrastructures on the performance of YouTube is investigated via a trace-driven simulation study. Section 6 presents related work. Finally, we conclude the paper in Section 7.

2. HOW YOUTUBE WORKS

YouTube is a web-based service that allows the sharing of videos via the Internet. Clients can upload their own videos and make them publicly available. Viewers can search for videos and then watch these videos on their computer or a mobile device. This section gives an overview on how the YouTube system works. We describe from the perspective of a client who requests a video from the YouTube web site with an emphasis on the communications between the client and YouTube servers. Figure 1 illustrates the communication between the client, YouTube server, and the CDN server. There are basically two different ways in which a client can request a video from YouTube: either by connecting directly to the YouTube web site, or by choosing an embedded video from a third party web page. In both cases videos are streamed as described in the following. When a client has chosen a specific video, an HTTP GET message is sent from the client to the web server. The following gives such an example.

```
GET /get_video?video_id=G_Y3y8escmA
```

This message indicates that the client is requesting a specific video which is identified via a unique video identifier, in this case *G_Y3y8escmA*. After receiving the GET message, the web server replies with an HTTP 303 See Other message, namely the redirection message:

```
Location: http://sjc-v110.sjc.youtube.com/get_video?video_id=G_Y3y8escmA
```

This message contains a location response-header field, which redirects the client to the video server from which the video will be streamed. The redirection mechanism introduces load balancing to the system since the web server can redirect the client to a video server in a dynamic way. Therefore, the web server has to have knowledge about which videos can be served from the video servers and the actual load of the video servers. To allow this kind of load balancing, YouTube makes use of a CDN service provided by both the YouTube video server farm and the Limelight CDN. Such a service automatically distributes content (in this case videos) to servers in the CDN. The goal of our work is not to identify the content distribution mechanism used by the CDN provider[‡]. We are rather interested in the implications on local distribution infrastructures based on the results of our measurement study. Flash Video[§] is the content format that is used for the video streaming from the CDN server to the client. The videos are encoded in a slight variation of the H.263 standard, which allows the player at the client to start rendering the video before the video file is completely downloaded from the server. For showing the videos at the client the Flash player is used which can be embedded in most web browsers.

3. MONITORING YOUTUBE TRAFFIC

In this section, we describe our methodology to monitor signaling and data traffic between clients in our campus network and YouTube servers. The methodology allows us to understand how a client gets a video stream from YouTube and then to obtain the YouTube usage statistics in our campus network. The monitoring process is a two-fold process. In the first step, we analyze signaling and data traffic, i.e., the HTTP message exchange between the client and the YouTube web server. Following the first step, we monitor video streaming traffic, i.e., the TCP headers of the video data sent from the CDN video servers.

3.1 The Monitoring Environment

We use the following equipment to monitor the traffic between clients in our campus network and YouTube servers. The measurement equipment is a commodity PC, installed with a DAG card[1] to capture packet headers. It is placed at the gateway router of UMass, Amherst, connected via optical splitters to the Giga-bit access links connecting the campus network to a commercial network. The TCP and IP headers of all the packets that traverse these links are captured by the DAG card, along with the current timestamp. In addition, we capture the HTTP headers of all the HTTP packets coming from www.youtube.com. Note that all the recorded IP addresses are anonymized [¶].

3.2 Monitoring Web Server Access

For each outgoing or incoming packet through the gateway router, its arrival time, size, source and destination IP addresses, and port numbers are recorded. If the destination or source IP address belongs to the pool of YouTube web servers, the HTTP header is recorded in addition. Initially, a client has to connect to the YouTube web site to search and then watch a video. The recorded information captures both the HTTP GET message and the HTTP 303 See Other message as described in the previous section. These messages allow us to extract information about the requested video, such as video identifier, when it is requested, and from which CDN server the data will be streamed. By tracing the video identifier, we gathered information about how often and at which points in time a certain video is requested from clients in our campus network. The CDN server location information allows us to trace the video streaming data which gives us additional information such as the amount of data that is sent from the CDN servers to the clients and the bandwidth that is consumed by these video streams.

Below, an HTTP messages obtained from a captured trace is shown. The first part shows the HTTP GET request from the client to the YouTube web server. This GET message contains a unique identifier for the video

[‡]Detailed information about the content management can be found at http://www.limelightnetworks.com

[§]http://www.adobe.com

[¶]We limit the scope of monitoring to such types of payloads, because of privacy issue and the limited computational and I/O capability of the monitoring equipment that needs to keep up with giga-bit speed.

that is requested by the client. In the case for the example shown below this unique identifier is **G_Y3y8escmA**. Monitoring the video identifier allows us to gather information about how often and at which points in time a certain video is requested from clients in our campus network.

```
GET /get_video?video_id=G_Y3y8escmA&t=OEgsToPDskKGx28kpTfHkYLmOQ-vE7KA
HTTP/1.1

Accept: */*
Referer:
http://www.youtube.com/player2.swf?video_id=G_Y3y8escmA&l=78&
t=OEgsToPDskKGx28kpTfHkYLmOQ-vE7KA&sk=MoQWS7wq_YYQ48Qf9mBTMAC
x-flash-version: 9,0,28,0
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.1;
.NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: www.youtube.com
Connection: Keep-Alive
Cookie:
```

The response to the HTTP GET message from the YouTube server (which is shown in the listing below) contains a message field that allows us to further analyze the actual data session between one of the CDN servers and the client. This field is necessary in order to obtain additional information such as the amount of data that is sent from the CDN servers to the clients, the bandwidth that is consumed for these video streams, and the locations of the CDN servers. As shown below, the YouTube web server replies to the HTTP GET message with a **HTTP 303 See Other** message. This message contains a location response-header field, which redirects the client to a CDN server from which the video content will be transmitted.

```
HTTP/1.1 303 See Other

Date: Wed, 04 Apr 2007 21:19:59 GMT
Server: Apache
Cache-Control: no-cache
Location: http://sjc-v110.sjc.youtube.com/get_video?video_id=G_Y3y8escmA
Keep-Alive: timeout=300
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

An extensive analysis of the data that was obtained through the monitoring process is presented in Section 4.5.

3.3 Monitoring Video Streams

For monitoring the actual video stream, only TCP and IP headers of packets are recorded, which are sufficient to analyze the duration of the streaming process, the required bandwidth, and the amount of data transmitted in a streaming session. As shown in Section 3.2 information about the location and the exact time when a video stream was requested can be obtained from the trace created by monitoring the client-YouTube server HTTP message exchanges. This information can be used to identify corresponding video stream flows in the overall trace (the trace that contains monitoring information of TCP header from all incoming and outgoing traffic at the monitoring node). For example, if we assume that the HTTP 303 See Other message was monitored and tagged with timestamp HH:MM:SS, then the TCP header trace can be parsed with the following filter:

```
Search for first flow with
Timestamp >= HH:MM:SS
IP dest   == client
IP source == CDN server as specified in the location-response header
```

We define a flow as sequence of packets with the same 5-tuple (source and destination IP addresses, source and destination port numbers, protocol number) where adjacent packets are not too far from each other (e.g., a 60 second maximum inter-arrival time between adjacent packets in the same flow). We classify packets in each trace into a flow according to the definition. The construction of flows along with the exact timestamps of the start and end time allows us to obtain the following statistics:

Table 1. YouTube Traffic Traces

Trace	Date	Length (Hours)	Per video stats			Request stats			# of unique clients
			Total	Single	Multi	Total	Single	Multi	
1	05/08 - 05/09	12	12955	77%	23%	18043	55%	45%	2127
2	05/22 - 05/25	72	23515	77%	23%	32971	55%	45%	2480
3	06/02 - 06/07	108	17183	77%	23%	24211	55%	45%	1547

- *Duration of a streaming session:* Based on this information, we compute the average duration of a streaming session between a CDN server and the client.
- *Average data rate:* This data is important in order to determine how much bandwidth is consumed by clients requesting video clips and the load that is consequently placed on the network.
- *Amount of transferred payload data:* The amount of transferred payload data reveals the size of the transmitted video clip. This data can be used to determine the average, maximum, and minimum size of the video clips offered by YouTube. Such information is important to determine design parameters for distribution architecture, e.g., to dimension the storage space of a CDN server or a cache.

These parameters, generated by analyzing the trace data, are not only important to analyze and characterize the actual YouTube system but are also necessary to compare the existing distribution approach with possible alternatives (a topic we address in Section 5).

4. MEASUREMENT RESULTS

In this section we analyze the trace data obtained as described in Section 3. The main goal of this analysis is to investigate the local popularity of YouTube video clips, the relationship between local and global popularity, and how time scale and user population impact the local popularity. In the remainder of this section, we will describe the three different traces that we have collected, analyze the local popularity distribution and compare it with the global popularity distribution, investigate the influence of time on the popularity, and investigate the popularity for individual clients. An analysis of the actual video streaming data is presented in Section 4.5.

4.1 Traces

Table 1 gives an overview of three different measurement traces. The first measurement (*Trace 1*) was a shorter scale measurement over a 12-hour time span. *Trace 2* and *Trace 3* are traces that result from multi-day measurements (3 and 4 days respectively). "Per video stats" shows specific information about the distinct videos that were requested during the monitoring period for each trace. The "Total" column under "per video stats" shows the total number of distinct videos that were requested during the trace period; "Single" shows the percentage of videos that were only requested once; and "Multi" the percentage of videos that were requested twice or more. "Request stats" shows information about each distinct request that was posted from a client in the campus network during the monitoring period. The "Total" column under "requests stats" shows the total number of requests; "Single" shows the percentage of requests that were made for a video only once during the trace period; and "Multi" is the percentage of requests that were made for video clips that were requested more than once. Compared to the other two traces, *Trace 1* presents measurement results from a shorter duration. This trace is used to investigate short term behavior with respect to request rate and popularity of video clips. We took two sets of such longer term measurements for the following reasons:

- *Trace 2* was collected during the last week of the semester. Due to the larger population of students on campus we expected a larger number of overall requests during the measurement period. The third trace *Trace 3* was collected during the first week of the summer break and we expected a lower number of overall requests because of the lower student population on campus. The results shown in the *Total requests* column of Table 1 confirms our assumptions. The goal of these two different measurements is to investigate the influence of different user populations on request rate and video clip popularity.

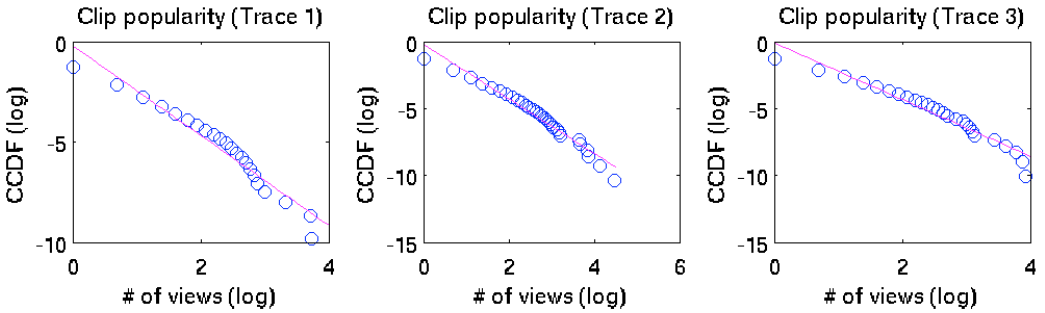

Figure 2. Video popularity for all three traces. X-axis is the numbers of requests and Y-axis is the log of CCDF.

- *Trace 3* was also collected with the goal of investigating how video content that is of local importance can have an influence on popularity and request rate. Shortly before that measurement was conducted, a video clip that shows part of the University's commencement was published on YouTube. Not surprisingly, this clip had the second highest popularity of all the clips requested during this measurement.

An interesting result of the three traces is the fact that all the statistics presented in Table 1 are essentially identical. That means, neither trace duration nor user population seem to have an influence on these statistics. The results also show that only less than 25% of all requested video clips are requested more than once.

4.2 Popularity

One focus of interest for video distribution systems is the popularity distribution of all video clips that have been requested by users in the access network. Obtaining the popularity information of these clips can give hints for the appropriate infrastructure for a video distribution system. For example, local caching at the access network level can be efficient if a subset of the requested videos have a high popularity. Figure 2 shows the complementary cumulative distribution function of the popularity (number of times a video is viewed) of video clips for all three traces. The complementary cumulative distribution (CCDF) of a random variable X is defined as $\bar{F}_X(x) = P(X \geq x)$.[2] The x-axis shows numbers of requests for video clips and the y-axis shows log of the CCDF. Comparing the results of all three traces shows that the popularity distribution is quite similar.

The results in Figure 2 also show that there is a high rate of video clips that have only been requested once during the measurement period (see also Table 1). To get a better insight into the request rate per video clip we determine this rate in relation to the overall request rate and show the results in Figure 3. For example, the leftmost bar of these three graphs shows the percentage of video clips that were only requested once during the measurement period. The second bar shows the percentage of video clips that were requested twice, and so on. Gaps along the x-axis indicate that no video clips were requested this exact number of times. These results show that a large number of videos are only requested once (77% in all the traces) while a smaller fraction (23%) are requested at least twice.

The efficiency of caching can be increased if popularity information about the objects to be cached is available. Unfortunately, this popularity information is neither available nor easy to obtain. Yet, with YouTube there is the possibility of obtaining a certain popularity metric. For each video clip the number of views since the clip was first published are given on the YouTube web site. In order to see if there is a correlation between the global YouTube popularity and our locally measured popularity we conducted the following experiment. For all of our three traces we obtained the global popularity information from the YouTube web site for each video that was requested locally. We then calculated the correlation coefficient between this global popularity and our local popularity:

$$\rho_{X,Y} = \frac{\sum((x-\bar{X})(y-\bar{Y}))}{\sqrt{\sum(x-\bar{X})^2 \sum(y-\bar{Y})^2}},$$

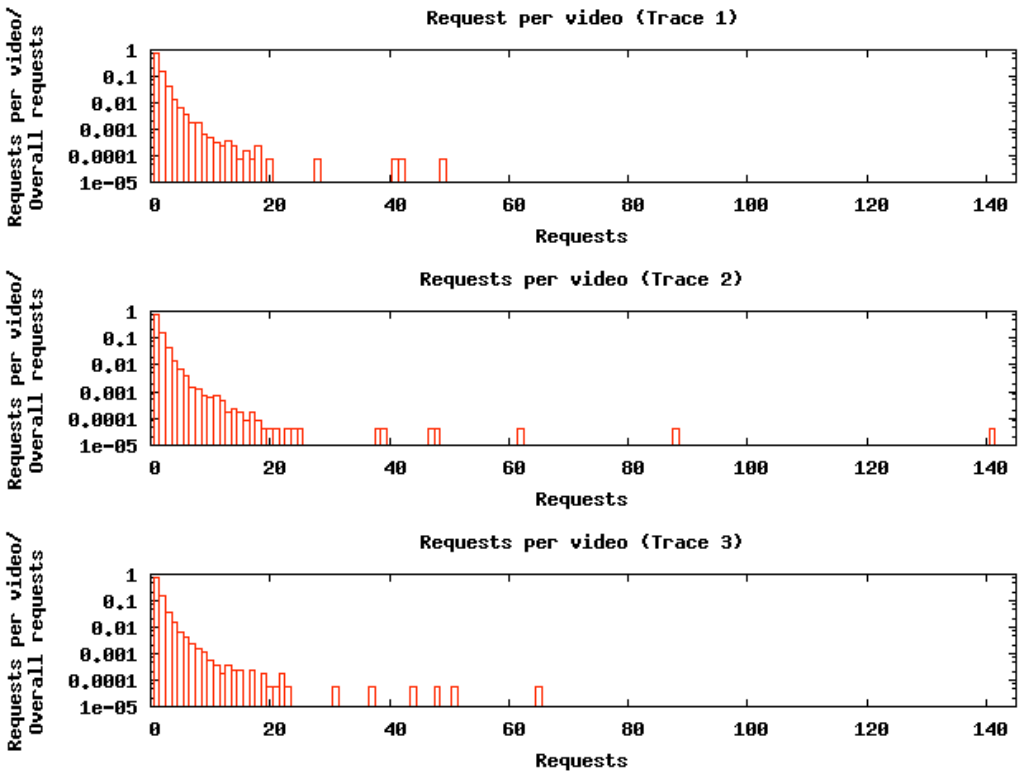

Figure 3. Rate of requests per video versus overall requests

where $\bar{X}$ is the sample mean of X. Our analysis shows that the correlation coefficient between both popularities is very low for all three traces, 0.04, 0.06, 0.06 for Trace 1, 2, and 3, respectively, indicating very weak linear dependency. Thus, we expect that using the global popularity information to make caching decisions will not improve its efficiency, e.g., increase the hit rate.

4.3 Request Rate over Time

In this section, we analyze the request rate for YouTube video clips over time. First we look at the binned request rate for all video clips which is shown in Figure 4. The figure shows the expected result that the request rate is high during the day and the evening until around midnight and then decreases significantly in the early morning hours. The request rate also drops slightly in the early evening hours (around 18:00) and increases again later in the evening (20:00). We conjecture that this effect is caused by the fact that the majority of students go for dinner during that time span.

Figure 5 shows the request rate per hour for the video clip with the highest popularity over Trace 2. Information about the clip that can be obtained from the YouTube web site reveals that the video was added to on May 23rd, basically the date when we started the monitoring process [‖]. Although the clip is a brand new addition to the YouTube video repository it becomes popular quite immediately and has the highest request rates during the first 12 hours of hour measurements. The popularity of that video clips seems to be very short lived, since the it is never requested during the last 16 hours of the measurements.

4.4 Requests per client

An analysis of the distribution of requests per client is presented in this section. The goal of this analysis is to investigate if clients sent multiple requests for video clips in a certain time period and if requests derive from a

[‖] Technically, we measured initial requests already in the last hour of May 22nd EDT. This is possible due to the fact that the time used to specify the *Added* information on YouTube might be set for a different time zone

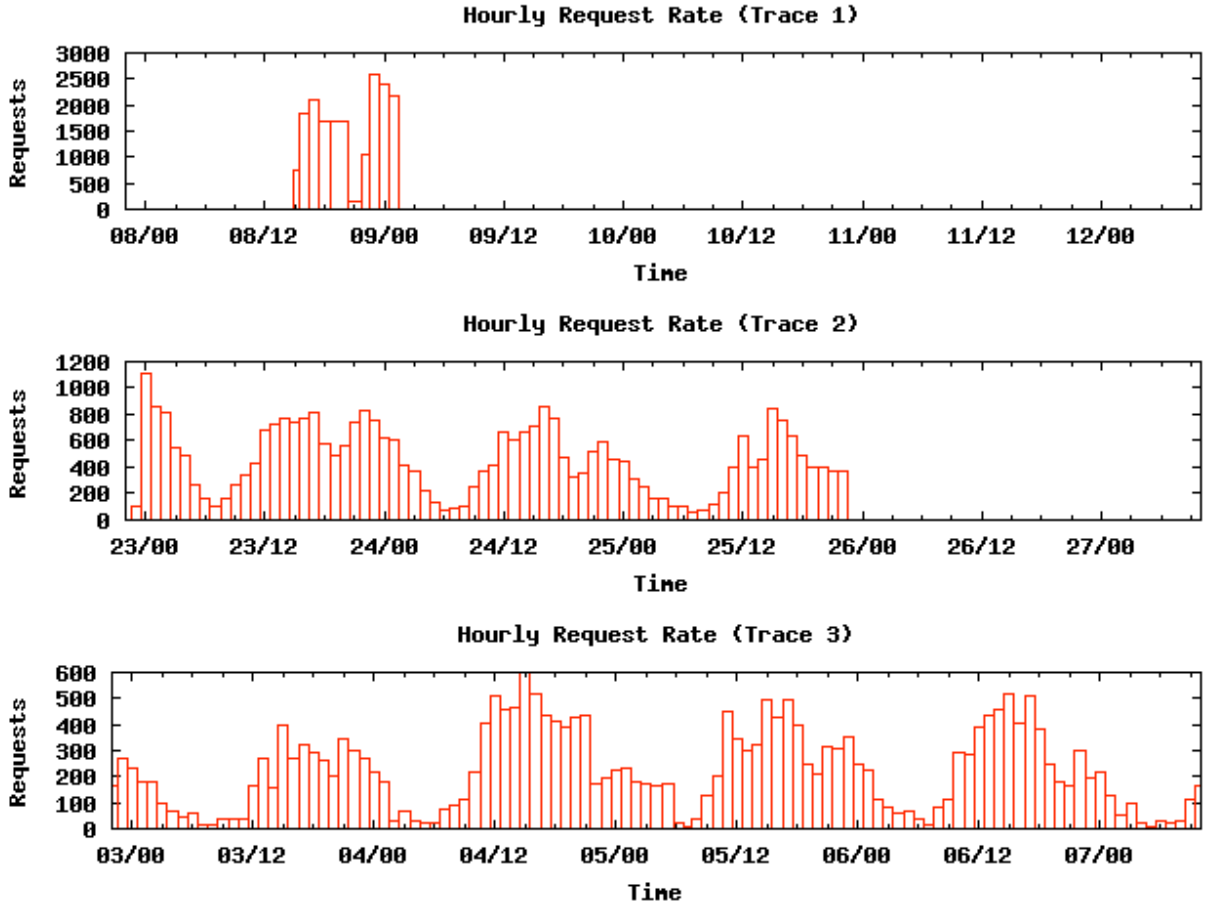

Figure 4. Hourly request rate for video clips

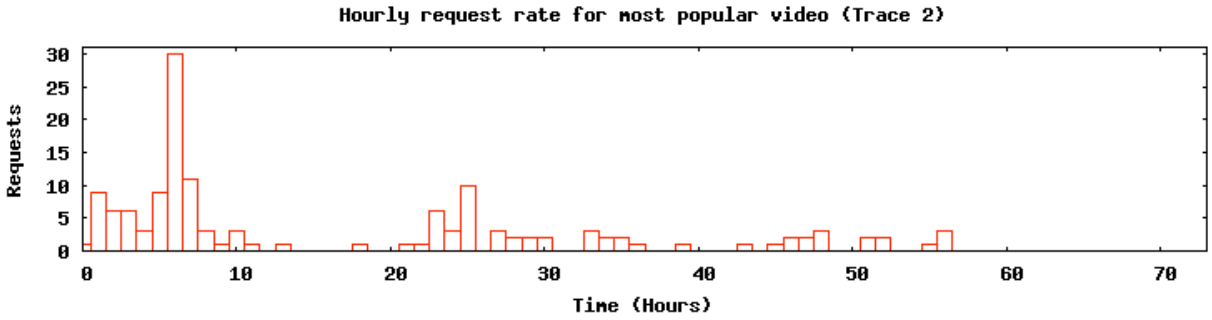

Figure 5. Hourly request rate for most popular video clip from Trace 2

larger group of clients or a smaller one. Table 1 shows that requests were made from 2127, 2480, 1547 different clients for Trace 1, 2, and 3, respectively **. Also here, the significant drop in clients for Trace 3 can be explained by the fact that this trace was collected during the first week of summer break. The scatter plots in Figure 6 show the distribution of number of requests over the overall client population. On the x-axis the number of requests

**The actual number of physical clients might be slightly higher due to the usage of DHCP for a few IP addresses in our campus network.

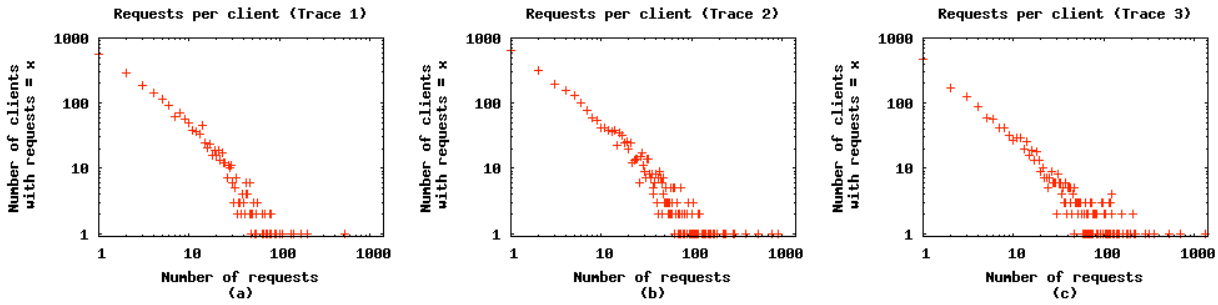

Figure 6. Distribution of requests per client

Table 2. Video clips requested twice or more from same client

Trace	Video clips with multiple requests from same client	Total number of requests	Max. number of requests per client	Avg. request inter arrival (seconds)	Median request inter arrival (seconds)
1	2149	3100	17	917	112
2	3899	5869	25	7998	231
3	3170	4893	47	10608	203

per client is shown and the y-axis shows the number of clients with x requests. For example, the Figure 6 (a) shows that 573 clients have issued only one request during the measurement period for Trace 1. All three graphs show a significant number of clients that issue two or more requests for video clips.

The results from this analysis encouraged us to look deeper in the request behavior of the clients. Thus, the traces were also analyzed to determine if the same client requests a video multiple times. The results (Table 2) show that a significant number of video clips are requested twice or more by the same client. This result is not necessarily surprising, since viewers might enjoy a video and watch it again or show it to others. The first column of the table shows the number of video clips that are requested more than once by the same client, while the second column shows the overall number of requests for video clips that are requested more than once by the same client. Note that a video that has been watched by a client is not cached in the client's local cache in general because of no caching option specified in the HTTP header sent by video servers. The third column shows the maximum number of requests for a single video clip requested by the same client. The median of the inter arrival times of requests for identical video clips is shown in the last column. The information presented here will be further used in Section 5, where different alternative distribution infrastructures for YouTube content are investigated.

4.5 Video Clip Traffic Analysis

In this subsection, we analyze transport-level connection that carries the content of a video clip requested by a YouTube client. Table 3 gives an overview on the results of the flow analysis for the three traces that were taken for this measurement. For each of the traces the table shows the duration of a data transport phase for a video stream, the number of packets per stream that were transmitted, the payload size per stream, and the average rate for the duration of a transport session. In addition, we also show the minimum and maximum values for these parameters. As shown in Table 3, the average duration of a flow is between 80 and 100 seconds. The longest flow had a duration of more than 4 hours. Since this result was rather surprising, we took a closer look at this flow and found that the average data rate for this flow was only $22Kbps$. It is beyond our means to determine the cause of this low rate. On the other hand there are actually videos with a duration of several hours. The minimum duration times (0.04 seconds) are also somewhat surprising at a first glance. But further research on YouTube revealed that there are video clips that consist of a single frame only. Transmission of that clip should be very short, since only a few packets might be needed for that small amount of payload. The column *Payload Size* shows the amount of payload that is transported per session. This includes data from

Table 3. Results from video stream flow analysis

Trace	Duration (seconds)			Packets			Payload Size (bytes)			Rate (Kbps)		
	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
1	99.62	4421.00	0.04	5202	149098	2	7.5×10^6	2.15×10^8	484	632	5450	0.54
2	95.81	2359.83	0.53	4478	89350	76	6.4×10^6	1.30×10^8	95760	646	8633	6.74
3	81.34	16956.28	0.04	4431	97452	2	6.3×10^6	1.42×10^8	452	908	10582	0.19

possible retransmissions due to packet losses. On average, the payload is on the order of several MB (between 6 and 7.1 MB). The final columns in Table 3 show that data was streamed with an average bandwidth between 630 and slightly more than $900KBps$. There are two possible reasons that contribute to these relatively high values. First, the university has a high bandwidth connection to the Internet and, second, all the clients are connected via either Ethernet or 802.11 b/g wireless network. Wang et al.[3] and Sripanidkulchai et al.[4] report much lower average data rates for clients that are also connected via DSL/Cable or modems.

5. DISTRIBUTION INFRASTRUCTURES

With the results we obtained in Section 4, the goal in this section is to investigate how alternative distribution infrastructure might impact the performance of YouTube. Three conceptually different alternatives for video distribution are investigated in a trace-driven simulation: local caching at the client, P2P, and proxy caching.

5.1 Local Caching at the Client

Local caching at the client can be compared with the process of local caching of web pages at the local browser's cache. Taking the results from Section 4.4 into account, which show that viewers watch the same video clip more than once, implementing such a process for YouTube videos could provide similar benefits for the viewer, i.e., startup delay and disruption of the video playout can be avoided if the video is already cached locally. We conducted a controlled experiment from a specific client where the same video clip was requested multiple times. Each time a new request was issued a new data stream was sent from one of the content servers. Thus, local caching is neither performed at the web browser nor at the Flash player. Default settings for web browser disk cache space are on the order of 50 MByte. Assuming that the video clip size is approximately the same as the payload measured in the flow analysis (Table 3 shows that the average payload size is around 7 MByte) several video clips could be cached in the browser's disk cache.

Table 2 shows that the average period for two consecutive requests for a video clip from one client is in the order of minutes (Trace 1) or hours (Trace 2 and 3). To investigate the impact of local caching at the client we ran a trace-based simulation using the data in traces 1, 2, and 3. Since each request for a video clip from YouTube is logged with a time stamp in the trace, we are able to generate a sequential (in time) list of requests for each client. In the simulation, we assume that each requested video is cached in the client's local cache. If there is not enough space to cache a new video, videos are removed in FIFO order to free up storage space in the cache. In addition, the request for this video is counted as a cache miss. In case the requested video is already in the cache, no further action is necessary and the request is counted as a cache hit. Figure 7 shows the result of this simulation. Here, we assumed that every video clip requires 7 MByte of storage space. The result of the simulation shows that a relatively small local cache can improve the overall performance of the system. For example, the comparison of a system without any caching and a system where each client has a 50 MByte cache shows that 3283 of the 3899 (video clips with multiple requests from same client shown in Table 2) could be served from the local cache of the clients (in the case of a simulation based on data from Trace 2).

5.2 Peer-to-Peer

Since the simulation results of a local caching approach show an improvement in the overall system performance, we were interested if the performance could be further improved by applying a P2P-based approach. Similar to the client-based caching described in Section 5.1 the client caches the video clip locally in the case of a cache

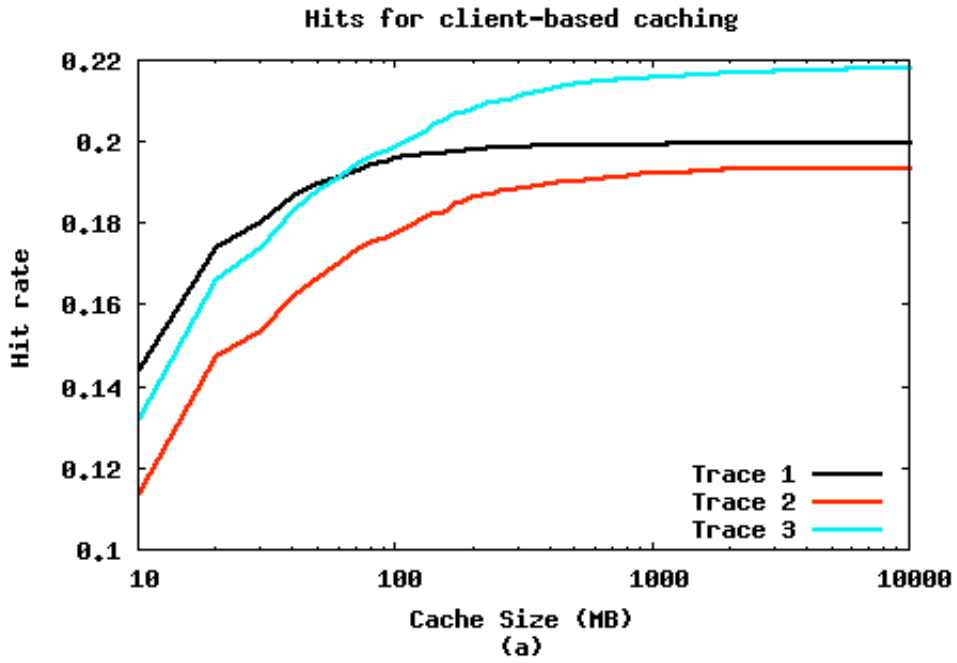

Figure 7. Overall sum of cache hits based on the local cache size

miss. But in P2P caching the client also checks if the requested clip is stored in the cache of another client in the campus network. In case of a cache hit the client can play the video clip from its own cache or retrieve it from another client in the campus network. The latter is only possible if one of the other clients in the local access network has the requested video clip in its local cache. To evaluate this P2P-based approach we ran another trace-based simulation. We assume that a requesting client knows (either via a central entity or a distributed hash table) if the requested content is stored at a local peer. In addition to the YouTube trace file we make additional use of the flow trace file (see Section 3) that contains information of TCP/IP headers from all incoming and outgoing traffic at the gateway. We use this additional trace file to gather information about the availability of a peer. It might be that a peer holds the requested video clip but is not online, which would result in a cache miss. In this case, the requesting client would have to retrieve the clip from one of the YouTube content servers.

For the P2P simulation, we assume that a client is online when data was sent from or to it. This information can be obtained from the flow trace file, since all the incoming and outgoing TCP headers are captured in this file. Clearly, a client can be online even if it is not sending or receiving any data. To account for this fact we run the simulation in different configurations. Each time a client makes a request for a video clip a lookup is performed to determine if the clip is already stored in another client's cache. Then the flow trace is parsed for this specific client's network activity, with a time window specified. For example, if this time window is set to 30 minutes it is assumed that the client was also active 15 minutes before and after the observed network activity from this client. The results of this simulation are shown in Figure 8. The x-axis represents the time window size mentioned above and the hit rate is shown on the y-axis. Figure 8 (b) shows the results of a slightly modified simulation. In this case, the window size is fixed to 30 minutes and the local cache storage size is varied between 10 MB and 10 GB. The results show that the hit rate can be significantly improved with a local storage size of 1 GB. Increasing the storage size beyond that size improves the hit rate only marginally. The comparison between the client-based and the P2P-based caching shows that the latter improves the overall system performance either marginally (Trace 2) or performs even worse (Trace 1 and Trace 3) than the client-based caching approach. The latter is a rather surprising result which is caused by the caching mechanism we assumed in the P2P case. Here, a client caches a video locally if it is not cached by another client in the access network. If this video is requested by another client it will either be served from the client which originally cached the video if it is online, or it will

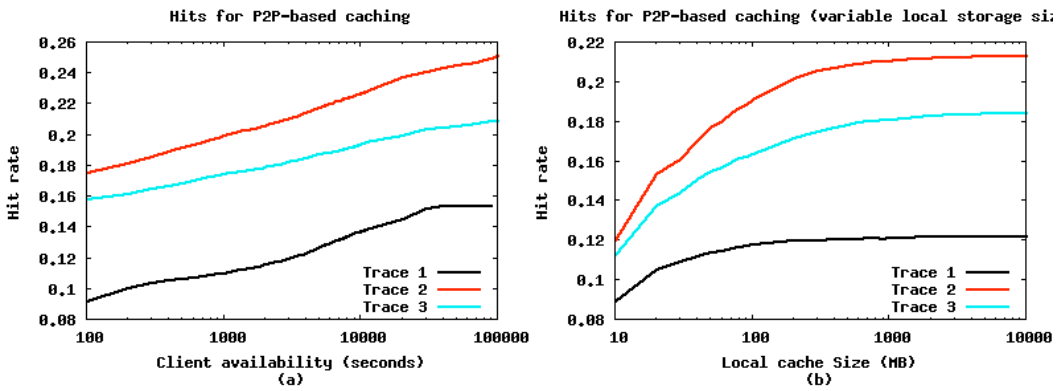

Figure 8. Cache hits for a P2P caching approach. The x-axis shows different intervals for peer availability.

be counted as a cache miss if the client is offline. In future work, we plan on investigating a modified caching scheme where videos will be cached locally if a client that already stores the video should not be online.

5.3 Proxy Caching

In this section, we assume that the there is one proxy cache that serves the University's access network. Client requests for YouTube content will be redirected to the proxy. Each request will be analyzed at the cache and the following actions are taken at the cache. In case the requested video clip is already stored on the proxy it will be directly transmitted from the proxy to the client. If the requested video clip is not stored on the proxy, the proxy can make two decisions. Based on its local caching strategy the proxy decides to cache the requested video clip. It then forwards its own request for the video clip (originally requested by the client) to the YouTube web server. Upon reception of the data stream the proxy will start storing the video payload on its local storage and also forwards the data to the client. If the proxy decides not to cache the requested video clip it will not interfere the message exchange between the client and the YouTube web server.

We simulated a cache-based content distribution approach for YouTube video clips where the total storage size of the cache located at the gateway is varied between 100 MByte and 150 Gbytes. The results of this simulation are shown in Figure 9. The x-axis of this figure shows the the storage space while the hit rate is shown on the y-axis. After a video clip is initially cached by the proxy all following requests for that clip are served by the proxy and counted as a cache hit. For cache replacement a simple strategy is employed in which the oldest cached content is removed to create space for new content. Each of the three slopes in the figure represents the result of a simulation that was run based on the data from Trace 1, 2, and 3, respectively. Here, we assume that the payload size determined by the video clip traffic analysis in Section 4.5 roughly reflects the size (in terms of storage needs) of the video clips. Figure 9 shows that a small increase in cache size (from 100 MB to 1 GB) increases the hit rate for all three cases by almost 10%. Increasing the storage size to 100GB will also the increase of the cache rate almost to the maximum. Note, the theoretical maximum hit rate is determined by the number of video clips that are requested at least twice. Since these numbers where derived by taking TCP payload into account one can assume that the actual storage space for the video-only data would be even smaller. Our approximation still includes overhead for possible additional headers and payload that might be caused by TCP retransmissions.

Overall, the results of the simulation show that a central proxy caching approach for a local access network with thousands of clients is an effective low-cost solution. Even standard desktop PCs today are equipped with the required storage space of several hundred Gbytes.

6. RELATED WORK

One major research area that is related to our work is concerned with the analysis and characterization of streaming services in the Internet. Early work in this area is focused on the characterization of videos on the

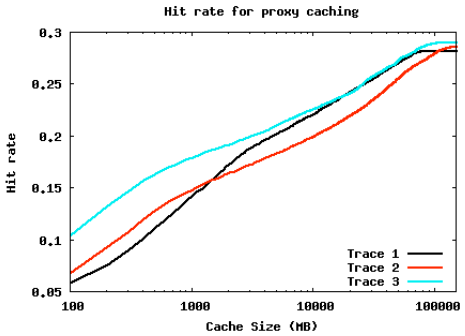

Figure 9. Hit rate based on proxy cache size

web[5] and how users access such videos.[6] Chesire et al.[7] apply a method very similar to our approach to measure streaming media workload in their university network. In their case the RTSP protocol is monitored to collect traces. These traces are then analyzed to obtain characteristics of the streaming media workload. In addition, the trace data is used as input to simulate a caching system in order to study the benefits of such an approach. An analysis of RealVideo performance with traces of clients from different geographical locations is presented by Wang et al.[3] Compared to the approaches presented above and our own approach presented in this paper, the authors make use of an active measurement method. Users were asked to use a modified version of the RealPlayer that monitors the requested video stream. Results from this investigation give hints about transport characteristics (available bandwidth and resulting quality of video) on the link between the video servers and clients. Another active measurement approach is presented by Li et al.[8] where a media crawler is used to retrieve audio and video clips from the web. Our approach is different since we monitor requests from all users in a university network to obtain information such as the local popularity of video clips. In addition, we collect low level information about the individual streaming sessions at the transport layer. Recently Silverston et al.[9] and Hei et al.[10,11] have reported their measurement study on popular P2P IPTV systems such as PPLIVE,[12] PPStream, SOPCast, and TVAnts. These works look at already existing live streaming P2P systems while we are interested in investigating whether a P2P architecture could improve the performance of an (almost) centralized system for pre-stored video.

Peer-to-peer networks for streaming video on the Internet have generated considerable interest recently.[12–15] Early work in this area was mainly focused on the efficient tree or mesh construction assuming simple bandwidth models. However, the quantitative analysis on the benefit of peer-to-peer streaming among peer nodes that are co-located in an access network has not been studied in those early research efforts. Only recently Huang et al.[16] showed the benefit of data sharing among peer nodes co-located under a same residential access network through analysis. On the other hand, in our paper, we analyze the benefit of peer-to-peer streaming by a trace-driven simulation based on the real YouTube trace. Huang et al.[17] have collected trace from a client-server VoD deployment for MSN Video and then conducted a trace-driven simulation to assess the amount of bandwidth reduction if a peer-assisted VoD service is implemented. Rather, we focus on the popularity of video and behavioral characteristics of clients served by YouTube service, which is known to be more popular than MSN Video service. In addition, unlike the approach used by Huang et al.,[17] we have collected YouTube traffic trace at the gateway of an large access network and conducted trace-driven simulations to evaluate the benefits of both client-based local caching and peer-to-peer network approaches. Also, we take into account the life time of clients by monitoring their network activity through the gateway router, which cannot be captured by the monitoring methodology used by Huang et al.[17]

7. CONCLUSION

In this paper, we have characterized the nature of YouTube traffic in a large university campus network. In particular, we have proposed a methodology to monitor YouTube signaling and data traffic between clients in our campus network and YouTube servers. By monitoring and correlating TCP/IP headers, and HTTP headers

of the packets exchanged between the clients and the YouTube servers at our university gateway, we analyzed the duration and the data rate of streaming sessions, the popularity of videos, and access patterns for video clips from the clients in the campus network.

Our results show that there exists no strong correlation between global and local popularity in YouTube videos. More specifically, we found that video clips of local interest have higher local popularity. Also, neither time scale nor user population is shown to have a significant impact on the local popularity distribution of the video clips served by YouTube. Somewhat surprisingly, we found out that many users watch a same video more than once. Since YouTube servers do not allow a user to cache the downloaded video in its local cache, this means that the same video must be downloaded whenever a request is made from the client. Also, only a few popular videos were requested by many users. These findings motivated us to look at the benefit of alternative distribution infrastructures to reduce the bandwidth requirements of YouTube service in our campus network. We used the trace collected from the measurement study to conduct trace-driven simulations to analyze and compare the potential bandwidth savings achieved by the alternative distribution infrastructures. The results of these simulations show that client-based local caching, P2P-based distribution, and proxy caching can reduce network traffic significantly and allow faster access to video clips. Our simulations show that most of the potential bandwidth savings come from caching and then serving video contents to users who watch same videos repeatedly.

As part of on-going work, we are currently evaluating a modified peer-to-peer distribution scheme that can benefit from caching video content served by other peers. Also, we plan to conduct a similar measurement study on other UGC services such as Google Video.

REFERENCES

1. "Endance DAG Network Monitoring Interface." http://www.endance.com.
2. W. Gong, Y. Liu, V. Misra, and D. Towsley, "Self-similarity and long range dependence on the internet: a second look at the evidence, origins and implications," *Computer Networks* **48**(3), pp. 377–399, 2005.
3. Y. Wang, M. Claypool, and Z. Zhu, "An Empirical Study of Realvideo Performance Across the Internet," in *Proceedings of ACM Internet Measurement Workshop, San Francisco, CA, USA*, pp. 295–309, Nov. 2001.
4. K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," in *Proceedings of ACM Internet measurement Conference(IMC), Taormina, Italy*, pp. 41–54, Oct. 2004.
5. S. Acharya and B. Smith, "Experiment to Characterize Videos Stored on the Web," in *Proceedings of SPIE/ACM MMCN, San Jose, CA, USA*, pp. 166–178, Jan. 1998.
6. S. Acharya, B. Smith, and P. Parnes, "Characterizing User Access To Videos On the World Wide Web," in *Proceedings of SPIE/ACM MMCN, San Jose, CA, USA*, pp. 130–141, SPIE, Jan. 2000.
7. M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming-Media Workload," in *Proceedings of USENIX Symposium on Internet Technologies and Systems, San Francisco, CA, USA*, Mar. 2001.
8. M. Li, M. Claypool, R. Kinicki, and J. Nichols, "Characteristics of Streaming Media Stored on the Web," *ACM Transactions on Internet Technology* , pp. 601–626, Nov. 2005.
9. T. Silverston and O. Fourmaux, "Measuring P2P IPTV Systems," in *Proceedings of ACM NOSSDAV, Urbana, IL, USA*, pp. 83–88, June 2007.
10. X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transaction on Multimedia* , 2007.
11. X. H. Y. Liu and K. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE Journal on Selected Areas in Communications* , 2007.
12. "http://www.pplive.com."
13. Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the Internet using an overlay multicast architecture," in *Proceedings of ACM SIGCOMM, San Diego, CA, USA*, Aug. 2001.
14. X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A data-driven overlay network for live media streaming," in *Proceedings of IEEE INFOCOM, Miami, FL, USA*, Mar. 2005.
15. A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proceedings of the IEEE Global Internet Symposium, Barcelona, Spain*, Apr. 2006.

16. Y. Huang, Z. Xias, Y. Chen, R. Jana, M. Rabinovich, and B. Wei, "When is P2P technology beneficial to IPTV service?," in *Proceedings of ACM NOSSDAV, Urbana, IL, USA*, June 2007.
17. C. Huang, J. Li, and K. Ross, "Can Internet Video-on-Demand be profitable?," in *Proceedings of the ACM SIGCOMM, Kyoto, Japan*, Aug. 2007.