Exploring YouTube's Content Distribution Network Through Distributed Application-Layer Measurements: A First View

Albert Rafetseder, Florian Metzger, David Stezenbach, and Kurt Tutschku
{albert.rafetseder, florian.metzger, david.stezenbach, kurt.tutschku}@univie.ac.at
Chair of Future Communication
Faculty of Computer Science
University of Vienna, Austria

***Abstract*—Content Distribution Networks serve a large share of today's Internet traffic from a multitude of locations, each in relative proximity to the respective consumers. In this paper, we analyze the performance of the YouTube video platform using *Seattle*, a distributed Internet testbed running on donated resources (like SETI@home), including equipment on the user's premises. This better reflects the experience seen by true users than dedicated test systems such as PlanetLab. In Seattle, measurements are restricted to the application layer, so we only read from and write into a stream (TCP) network socket, and perform DNS lookups. Using forty vantage points in different geographic regions running for more than 600 hours, we continuously measure the number of IP addresses www.youtube.com resolves to, and approximate the latency to actual video cache servers. We also monitor the latency between the vantage points, and estimate the packet loss. The methodology presented in this paper shows how insights even into large, distributed content delivery systems can be gathered with reasonable effort. The actual results can be of interest to e.g. network operators trying to improve the interworking with CDNs by studying their day-to-day operations for the good of the end user.**

I. Introduction

The topology of the Internet is constantly evolving, and so are the techniques used to provide content to consumers. Steadily increasing access and core network bandwidths both in the fixed-line and mobile domains have enabled simultaneous unicast multimedia streaming to millions of consumers. Content Distribution Networks (CDNs) are the current architecture of choice to implement this and many other types of large-scale services.

We focus on the YouTube CDN. Through active application-layer measurements we gain insight on the development of the latency to the cache servers and DNS-based load balancing. The measurements run on the *Seattle* distributed Internet testbed [1], [2]. Results are gathered on heterogeneous machines (ranging from servers at universities to desktop PCs and smartphones) in different timezones on end user's Internet connections. This lets us see the network from an end user's point of view, and accordingly experience (or miss) network quality. In future, fixed-line and mobile Internet Service Providers might want to go beyond understanding the methods unilaterally employed by CDN operators, and coordinate network control with them for optimization purposes.

The rest of this paper is structured as follows. Section II reviews related work. Section III discusses the evolution of the Internet's topology from client-server to current content distribution architectures. In Section IV, the Seattle distributed Internet testbed is introduced. Section V describes the methodology employed to gather our measurements. Section VI presents the results of the distributed measurement campaign, and Section VII concludes the paper.

II. Related Work

There already exists a number of publications related to YouTube, many of which consider the user behavior, or current trends among video content, often taking into account specific user groups such as students [3]. On the other hand, papers focusing on network-related issues seem to be less abundantly available. A selection of latency and throughput measurements are available in [4] for YouTube, Google Video, and Yahoo Video. One more recent publication developed a methodology to investigate data-center driven content distribution [5] to which our own methodology relates as well. Another notable paper [6] discusses YouTube traffic statistics collected at a Tier-1 PoP. As the authors note in the appendix to their paper, YouTube has changed the structure of their delivery system since the traces were collected. We will try to reveal that new structure in the course of this paper.

The most recent publication available to us [7] focuses on the server selection strategy employed by YouTube. The authors conclude from their traces from the edge of five networks that YouTube maps users to data centers based on a smallest-latency policy, and point out reasons for non-preferred accesses.

III. Architectural Evolution

Initially, the World Wide Web was constructed rather statically. Similar to remote login sessions that dominated the Internet before, web sites were each served from a single server placed on hosting sites connected to the Internet. To reach this site a user's request had to cross not only his ISP's domain but also often numerous transit domains between the user and the server.

Nowadays, the appearance of the Web has changed drastically. Over the past years, access bandwidths of wired and wireless Internet connections have been steadily increasing. This enabled new concepts in the Web. Pages are today often set up dynamically using server- as well as client-side scripting languages and extensively employ multimedia content. As a result, the demand on the servers increased up to the point that no single server could serve all request to a web site anymore.

This led to the development of various load balancing measures. In the early days of content distribution, downloads (e.g. large software packages) were predistributed to a set of mirror servers. Any interested user could now manually choose a mirror closest to his location and download from that server.

Content Delivery Networks (CDN), such as Akamai [8], Limelight [9], or the Google network, which have shifted into focus in the recent years, evolved from this mechanism. By setting up dedicated DNS servers in their domain and evaluating the IP address of the requesting users' DNS resolver, a mirror server is selected by factoring in geographical closeness, costs, load, or other factors. Content in the CDN is replicated to data storage in relative closeness to or even inside of access networks of Internet Service Providers. Thus, the load on carrier networks is reduced.

To fulfill the purpose of content delivery well and serve content consumers with acceptable performance even when there is high demand, those networks have to spread out into or near almost every ISP. This resulted in the emergence of huge overlay networks and increasing portions of Internet traffic originating from inside them. For example, Google, YouTube's parent company, now carries over 6% of the total Internet traffic [10]. The topology of these networks is however largely unknown to the Internet's BGP routing. This could pose difficulties to predicting or managing traffic volumes and changes thereof stemming from CDNs.

In future one could envision a cooperation between major sources and sinks of traffic, i.e. CDNs and ISPs, to collectively manage routes and traffic volumes.

A. *CDNs and DNS*

The Domain Name System forms the basis of all web-based CDN systems. For a regular website, the client asks its responsible DNS server (which is usually located in the ISP's domain) to resolve the host name to an IP address. To achieve this, the DNS server recursively queries the name servers responsible for the domain name until an address is finally retrieved [11] and transmitted to the client. CDN operators manipulate their name servers to deliver addresses in geographical closeness by factoring in the location of the request's source address, and answering with multiple address to improve the load balancing. Moreover, the time-to-live (TTL) of the name entries is intentionally kept short to be able to quickly react to changing demands [12], e.g. 300 seconds in YouTube's case. This practice is also criticized as it could negatively affect the performance of DNS [13].

IV. The Seattle Internet Testbed

Due to the distributed, large-scale nature of the YouTube CDN to be measured, we consider a distributed measurement facility. Our platform of choice is the *Seattle* Internet testbed developed at the University of Washington [1], [2]. Seattle has a number of properties that sets it apart from other testbeds and platforms:

Diversity of nodes. In contrast to systems like PlanetLab, Seattle does not require dedicated machines with good network connectivity, but is designed to run as a background process on devices such as smartphones and desktop PCs on heterogeneous, possibly mobile Internet connections. It therefore better reflects load and network conditions as experienced by end users.

Donated resources. For each node a user contributes to the platform, she gains the rights to run experiments on ten additional remote nodes in exchange. This allows for a great leverage for experiments, and encourages users to donate computational resources to the system.

High level of abstraction. Seattle executes programs in a programming language virtual machine based on a subset of the Python scripting language. The core API consists of merely 31 functions, making it easy to learn the language. Experiments are quickly prototyped and deployed on a broad scale.

Safety through restricted access. In order to ensure safety for the donors of resources, no low-level access is provided: Experimenters cannot access arbitrary files, only the send and receive calls of network sockets are exposed, protocols are limited to UDP and TCP. Furthermore, network bandwidth and CPU consumption are limited. In the following section, we will describe how to perform measurements of YouTube down to the network layer despite these restrictions.

V. Methodology

As indicated above, we deploy our measurement scripts on the Seattle platform. Initially, forty nodes in eight different time zones participate in the measurement campaign. Since there is considerable regional overlap between these vantage points, and we neither want to put too much burden on the donors' resources nor YouTube's, the measurements pause for a random amount of time chosen from the uniformly distributed interval $[0, 1800]$ seconds. Over all nodes, this yields one measurement every 22.5 seconds on average. We gather a number of metrics discussed below.

A. *Domain Name Mapping*

We knew beforehand from casual observations that YouTube employs a set of different domain names. To learn about the structure of hostnames YouTube uses, we download and parse the HTML code served by the website manually. It is seen that a number of different subdomains for different types of content exist. Through forward and reverse domain name resolution, we then map these names to IP addresses and back again, uncovering a region-based names' hierarchy that YouTube employs.

B. *Frontend Servers and Server Diversity*

In the previous step, we noticed that `www.youtube.com` (which we came to call *frontend*) resolves to multiple IP addresses, and that these addresses change over time. Therefore, we analyze on the vantage points which local frontend IP addresses are currently visible using Seattle's `gethostbyname_ex()` call which queries the node's default DNS resolver. The same lookup is performed for a video cache server from which we download a video.

We refer to the number of different IP addresses seen on one node at a particular instance in time as the *frontend server diversity*.

C. *Latency Measurement*

On the Seattle platform, the Internet Control Message Protocol (ICMP), and thus `ping` is not available to experimenters to determine the latency between a vantage point and a video cache server. The TCP three-way handshake [14] (including the API overhead) is timed instead.

A TCP connection is initiated by one host sending a segment with the SYN (synchronize) control bit set. Upon this, the other host answers with a segment that has the ACK (to acknowledge the initial segment) and SYN bits set. When the initiator of the connection times the interval between sending SYN and receiving ACK, SYN, she then knows the latency between her and the remote host [15]. To complete the three-way handshake, a third message is sent to acknowledge the second segment, so the remote side can also measure the latency.

From the latency values between our vantage points and the respective video cache servers, we cannot know whether the latency variations occur in YouTube's network or close to the vantage point. We therefore also measure the latency between all vantage points. If the latency from a given node to all destinations varies at a given instance of time, we then assume the root cause lies in networks close to that node. If on the other hand there is no such synchronism, one single host or cache server will stand out.

The TCP three-way handshake only works if the contacted host is willing to open a connection on the used port. It should answer with an RST (reset) segment otherwise. We know that the Seattle's Node Manager port 1224 is open on all vantage points so that experimenters can contact the nodes to upload programs, download data, query the node status etc. For YouTube's servers, we use the well-known HTTP port 80.

D. *Loss Estimation*

TCP implements reliable transmission between two hosts. If the connection is already set up, segment loss is identified by receiving duplicate acknowledgements from the receiver, or by expiration of the retransmission timer, whose timeout is set to track the current round-trip time of the connection. If the connection is in the process of being set up, no round-trip time is known, and the initial retransmission timeout (RTO) is set to three seconds [16]. We can exploit this fact and count the number of outliers among the latency values to estimate the packet loss.

E. *Sequence of Measurements*

The sequence of measurements carried out at each node is as follows. First, the latency to the other vantage points is measured by timing four connection attempts to a known open TCP port. Then, the frontend domain name is resolved. A website including the URL to the actual video cache server is requested from the first IP address returned from the DNS. The HTTP headers and the website are parsed, yielding an HTTP cookie and the final URL. At last, the video cache's domain name is resolved to an IP address, and the latency to the server is determined. Each step is logged to a file for offline analysis.

Apart from the latency measurements between all vantage points, the steps performed are identical to what a user agent (web browser) would do when a user visits YouTube's website and watches a video.

VI. Results

Below we report on the results of our distributed measurement campaign. The experiment ran on 40 Seattle nodes for 26 days[1], and performed more than 5 million latency measurements and 70,000 DNS lookups.

A. *Domain Names and IP Addresses*

We categorize the domain names seen in URLs on YouTube's websites as follows: Unique global host names, video cache host names, static cache host names, and unknown types.

Global host names. Each YouTube server IP address has one global unique entry in Google's DNS servers. They are encoded in the form of $\mathtt{r}\alpha.\beta\gamma\mathtt{s}\delta\mathtt{.c.youtube.com}$ where α is 1-24, β is an International Air Transport Association (IATA) airport codes [17], γ is 01-16, and δ is 01-12. Note that not all global host names can be resolved at all times. In one test run we found 29 different IATA codes in the global hostnames, yielding around 3,200 names total.

Video cache host names. Names of this type are used to serve the actual video content to the user. A single video can be served from different video caches. The resolved IP addresses are specific to the region the user request originates from. The name format is $\mathtt{v}\alpha\mathtt{.lscache}\epsilon\mathtt{.c.youtube.com}$ where ϵ is 1-8. We were able to map the video cache host name to the global host names discovered during one test run most of the time

Static cache host names. These are used to host static content of the website, e.g. style sheets, images, script files, or the YouTube Flash player. As there is just a single subdomain, only one set of addresses per region is responsible for this type of content. Names of this type take the form of **`s.ytimg.com`** and **`i.ytimg.com`**.

Several other types of domain names in URLs can be observed, but they are not used frequently and their use was not investigated further.

[1]starting on Tuesday, January 4th, 2011

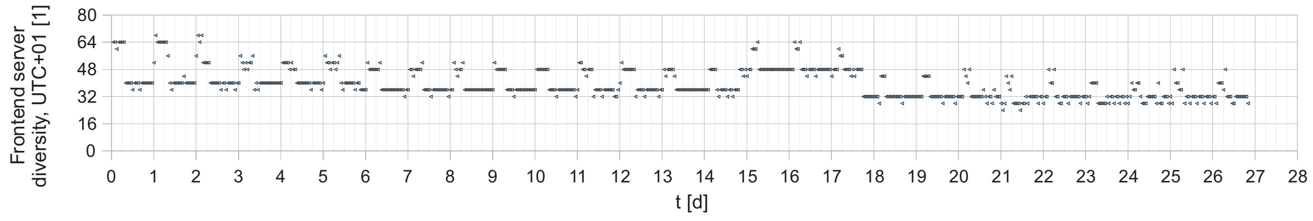

Fig. 1. Number of frontend servers for Central European nodes

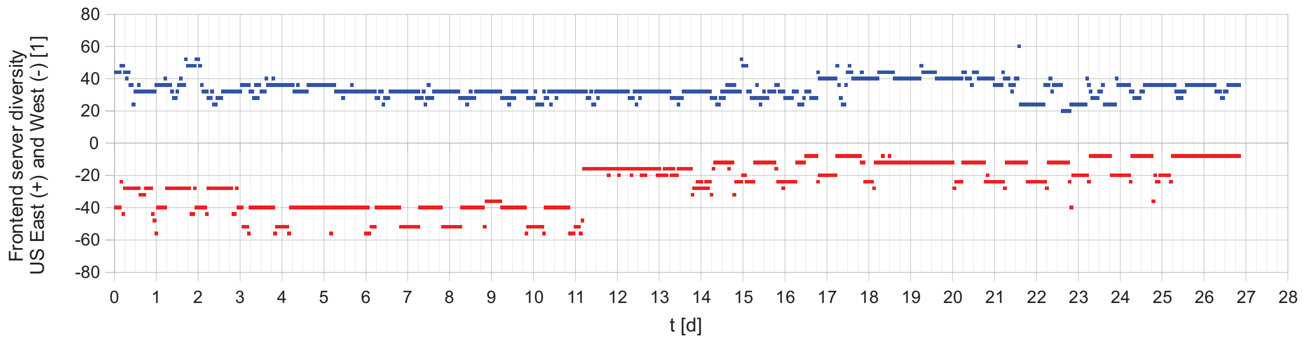

Fig. 2. Number of frontend servers for East and West US nodes

B. Frontend Server Diversity

As discussed above, there are globally unique host names and host names resolving to regionally adapted IP addresses. The WWW frontend, `www.youtube.com`, is an example for the latter. Overall, we found 356 different frontend IP addresses during our measurements. Not every address is available at all times. Figure 1 shows the number of available frontend IP addresses over time for those vantage points in the UTC+01 timezone, and Figure 2 has the same information for U.S. nodes. Counts for UTC-05 and -06, i.e. Eastern and Central timezones, are shown as positive, UTC-07 (Mountain) and -08 (Pacific) as negative. The readings per timezone are grouped by the hour of measurement.

Note the diurnal pattern in the results for each timezone. Given that the main x grid in the plots denotes midnight UTC+01 in all Figures, the local frontend server diversity is highest in the early hours of the day for the Central European nodes, dips around midnight (local time) for the Eastern part of the U.S., and has a rather pronounced twelve-hour pattern with the low values during the afternoon (local time) in the Western part of the U.S.

To understand those troughs and peaks in frontend server diversity that last for several days in a row, please consider Figures 3 and 4. For these plots, the frontend IP addresses were numbered sequentially, starting with the address occurring most often. Where Figure 1 shows for example an increase in frontend server diversity on day 15, followed by a return to the previous base level three days later, the details in Figure 3 reveal that a new IP address block is introduced in parallel to the existing one, and later takes over the duties completely. Furthermore, a number of blocks with short uptimes are identified. These probably serve the peak load. Conversely, some nodes are seen to vanish at the same instances of time. We could only speculate on the operational necessities for this control strategy.

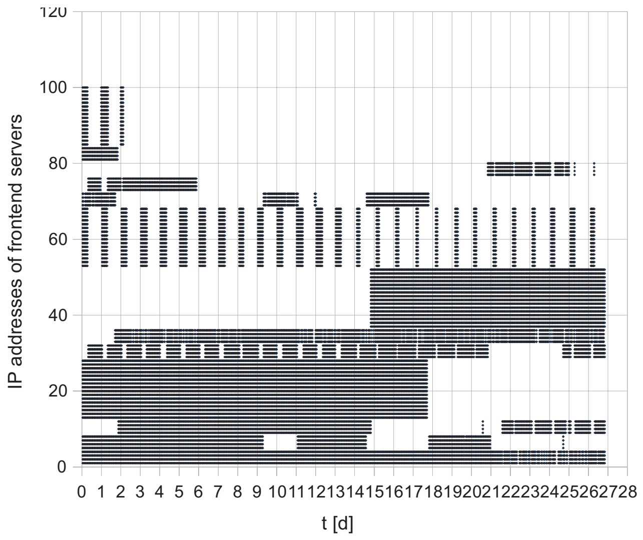

Fig. 3. Frontend servers for Central Europe

C. Latency to Video Caches

The latency to the video cache server currently designated by the CDN is subject to high variations. As an illustrative example, consider the case of one Seattle node in Chicago, Illinois (although similar incidents are seen on other nodes as well). Figure 5 plots the latency between this node and

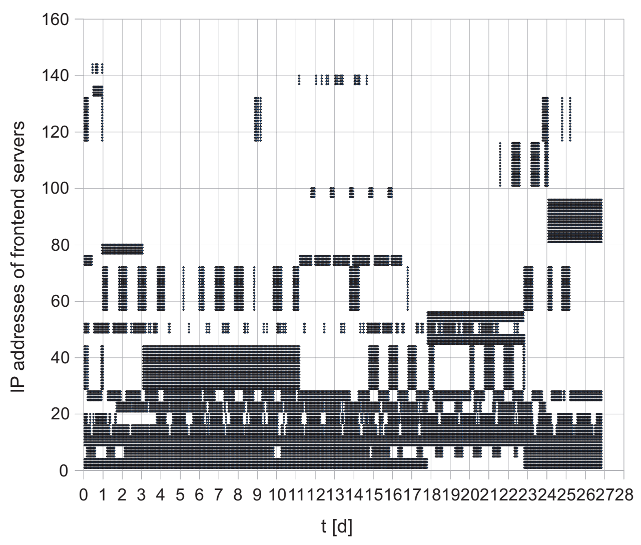

Fig. 4. Frontend servers for East and West US nodes

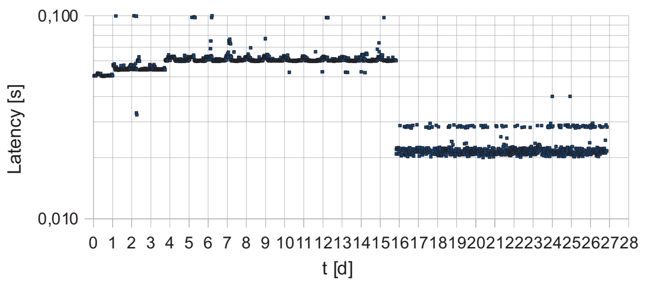

Fig. 5. Latency to the current video cache server for one vantage point

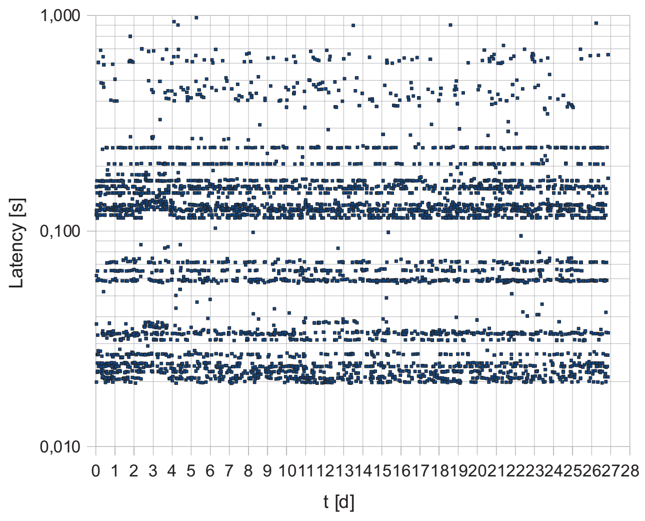

Fig. 6. Latency to all other vantage points

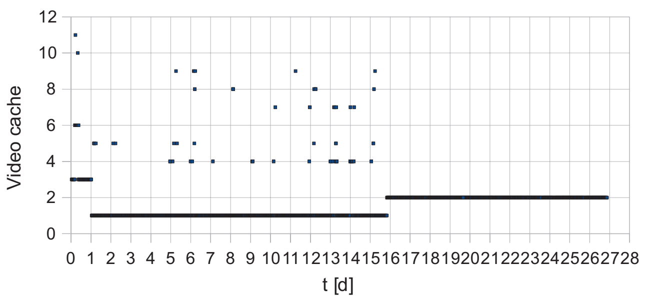

Fig. 7. Currently available video cache servers for one vantage point

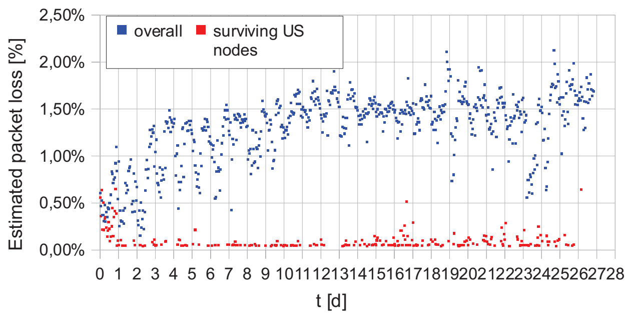

Fig. 8. Estimated packet loss

whatever video cache server was identified in the HTML code of YouTube's home page at the time of measurement. During the first four days, the latency increases stepwise. It then stays at a base level of 60ms but exhibits diurnal variations. On day 16, it suddenly decreases to one third and stays at this level for the rest of the measurements, save for a number of outliers.

To find out if this is a result of the node's or YouTube's network changing, compare Figure 6. The latency to all other vantage points remains unchanged on day 16. On the other hand, a small deviation not seen in the previous plot is found here at midnight (UTC+01) on day 3, but this does not affect all vantage points.

It stands to reason that the improvement in latency to the video cache server lies within YouTube. Analyzing the video cache server IP addresses that were resolved just before the latency measurements (see Figure 7), we see that shortly before day 15 ended, the test video indeed started to be served from another server.

D. Estimated Packet Loss

The packet loss is estimated from the number of large outliers (greater than 2.4 seconds) found among the latency values. Of the $n = 5,319,954$ latency measurements between the vessels, $l = 134,370$ were outliers. Each measurement consists of two segments (SYN–ACK, SYN), and outliers are assumed to cause resending of one of the two segments, yielding an overall number of packets of $2n + l$, of which l were lost. The overall packet loss ratio is thus estimated to be 1.25%.

However, the estimated loss ratio is not constant over the time of our measurement. Figure 8 shows patterns on different timescales. First, there is a diurnal pattern again. Second, the estimated loss increases during the first ten days of the experiment. This might be a result of New Year's vacations ending, and people returning to work, thus increasing the overall network load.

As a result of the Seattle nodes running on university machines as well as home user hardware, the number of nodes participating in the experiment diminishes over time. We therefore estimated the loss experienced by stable nodes. Figure 8 also plots the estimated loss only for those North American nodes that "survived" (were online for the whole duration of) the experiment. The baseline is much lower now (0.05%), with the mean at 0.1%. We conjecture that this large difference to the overall estimated loss can be explained with the surviving nodes being university and research machines on network links with better quality than the typical end user's.

VII. Conclusion and Outlook

We sketch in this paper how content distribution across the Internet has changed from a single-instance single-location multiple-transit architecture to highly distributed Content Distribution Networks located close to the users' access networks. We describe our distributed measurement campaign (using the *Seattle* platform [1], [2]) through which we analyze the structure and mapping to IP addresses of YouTube domain names, count the number of concurrently visible IP addresses for the frontend servers, explain temporal variations, measure the round-trip time between our vantage points and cache servers, pinpoint the cause of latency improvements through comparison of latency and IP diversity metrics, and estimate the packet loss for the participating nodes. Note that all measurements are performed on the application layer.

The outcome of our work is twofold: On one side, we present a methodology enabling interested parties to take a user's perspective on the performance of large-scale, highly distributed network systems. Our methods can be implemented without a need for specialized measurement equipment, access to corporate infrastructure such as DSLAMs, confidential data, etc. On the other side, the measurement results reveal operational procedures in YouTube's CDN. We think that ISPs and mobile operators could use such information to improve the cooperation between their networks.

For the future, we plan to look into the role and value of the global host names for geo-locating servers and traffic flows. Furthermore, we are interested in the download behavior of the YouTube player, and want to experiment with additional distributed application-layer based measurement methods such as packet-pair bandwidth estimation techniques [18].

Acknowledgments

The authors would like to thank Oliver Michel, Ákos Lukovics, and Bernhard Gruber for their investigation of YouTube's names structure, data mining, and visualization.

References

[1] The Seattle Internet Testbed. [Online]. Available: https://seattle.cs.washington.edu/

[2] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *Proceedings of the 40th ACM technical symposium on Computer science education*, ser. SIGCSE '09. New York, NY, USA: ACM, 2009, pp. 111–115. [Online]. Available: http://doi.acm.org/10.1145/1508865.1508905

[3] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298310

[4] A. Banerjee and D. Barman, "Characterizing quality of content distribution from YouTube like portals," in *Proceedings of the 2007 ACM CoNEXT conference*, ser. CoNEXT '07. New York, NY, USA: ACM, 2007, pp. 56:1–56:2. [Online]. Available: http://doi.acm.org/10.1145/1364654.1364719

[5] V. K. Adhikari, S. Jain, G. Ranjan, and Z.-L. Zhang, "Understanding data-center driven content distribution," in *Proceedings of the ACM CoNEXT Student Workshop*, ser. CoNEXT '10 Student Workshop. New York, NY, USA: ACM, 2010, pp. 24:1–24:2. [Online]. Available: http://doi.acm.org/10.1145/1921206.1921232

[6] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "YouTube traffic dynamics and its interplay with a tier-1 ISP: an ISP perspective," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 431–443. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879197

[7] R. Torres, A. Finamore, J. Kim, M. Mellia, M. M. Munafò, and S. Rao, "Dissecting Video Server Selection Strategies in the YouTube CDN," *Purdue University e-Pubs*, 2011. [Online]. Available: {http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1407\&context=ecetr}

[8] Akamai. [Online]. Available: http://www.akamai.com/

[9] Content Delivery Network — Limelight Networks. [Online]. Available: http://www.limelightnetworks.com/

[10] "Google now second-largest ISP, carries 6.4% of Internet traffic." [Online]. Available: http://www.networkworld.com/news/2010/102710-google-number-two-internet-traffic.html

[11] P. Mockapetris, "Domain names - implementation and specification," RFC 1035 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[12] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms," RFC 3568 (Informational), Internet Engineering Task Force, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3568.txt

[13] P. Vixie, "What DNS is not," *Communications of the ACM*, vol. 52, no. 12, pp. 53–47, 2009.

[14] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[15] Nmap – free security scanner for network exploration & security audits. [Online]. Available: http://nmap.org/book/man-host-discovery.html

[16] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988 (Proposed Standard), Internet Engineering Task Force, Nov. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2988.txt

[17] "IATA Airline Codes Search," Resources available online at http://www.iata.org/ps/publications/Pages/code-search.aspx/.

[18] J. Strauss, D. Katabi, and F. Kaashoek, "Measurement Study of Available Bandwidth Estimation Tools," *ACM Internet Measurement Conference*, 2003.