Analyzing the video delivery strategies of video publishers

Georgios Papadimitriou, Zahaib Akhtar(Mentor)
CSCI-651 Advanced Computer Communications

1. INTRODUCTION

Video providers such as YouTube and Netflix are well understood today. This is because of the attention they have received due to a large user base. A number of research studies have explored their various aspects including network architecture, user engagement and video delivery mechanism etc. However, Internet video ecosystem consists of a large number of small scale video publishers which haven't received much attention. While the bulk of the video traffic today is formed by big providers, these small scale providers can be considered the tail of the video traffic. The end goal of this project is to analyze the video delivery strategies of these smaller providers and compare and contrast them with popular providers.

Importance. This project is interesting to the field, because it can provide insight to the technologies used for video content delivery by small providers, where video plays a supporting role in their businesses, such as news outlets or online learning platforms. Additionally by analyzing and comparing these data with what major providers are doing, such as YouTube or Netflix, we can provide suggestions that will benefit the smaller ones. Finally, from this project I will get a good understanding of how the video content delivery pipeline works, and familiarize myself with data analysis tools.

Challenges. First, video delivery technologies differ between providers, because they are trying to match them to their own needs and delivery strategies. Due to this, we need to discover and specify a common way of retrieving metadata, relevant to the delivered content, so we can automate the acquisition process. Second, in order to provide useful and reliable insights to the video delivery strategies we need to acquire data for multiple videos and from as many video providers as possible. This identifies the scale of the suggested analysis, and could be the most challenging part of this project.

2. BACKGROUND

In order to serve video, publishers typically use a *Streaming Protocol*. There are atleast four well known streaming protocols which differ in popularity. These include 1. HTTP Live Streaming (HLS) [5] from Apple, 2. MPEG-DASH which is the Industry standard protocol developed by the Motion Picture Experts Group (MPEG) [9], 3. HTTP Dynamic Streaming Protocol (HDS) [4] implemented by Adobe and 4. The Smooth Streaming Protocol [17] which is implemented by Microsoft.

These streaming protocols specify a number of different characteristics which are vital to video delivery. These include the network encapsulation format for the video chunks, the set of available bitrates, the duration of an individual chunk and the URL to fetch the video content from. These are specified in what is called a *Manifest file*. Before a video player start playback, it downloads the manifest file which informs the player about these attributes hence enabling it to request video content and intelligently adapt bitrates based on the network throughput.

Video publishers use a number of different strategies to serve video content. Their strategies can vary along a number of dimensions including the choice of streaming protocol, the number of available bitrate levels for adaptation, chunk duration and the content distribution network (CDN). Their diversity along these dimensions can be understood by analyzing the manifest files. We next describe our methodology for selecting a candidate set of publishers and obtaining video manifest files for their video content.

3. METHODOLOGY

To bootstrap our measurement study we first need a list of content publishers that serve video. Second, we need a way to crawl through these selected publishers' websites to access the manifest files associated with their video content.

3.1 Selecting Video Publishers to study

We collect a set of video publishers of interest by going through Alexa top 500 website list [15]. Alexa provides a ranking of websites based on the traffic, it also provides a ranking of websites based on the type of content they serve. To pick popular video publishers we select websites from the "Sports" and "News" category. These categories are important because most websites in this category primarily serve video content. We pick a total of 25 websites.

3.2 Automated crawling and data collection

We use and extend a number of open source tools to build a pipeline that enables automated collection of the video manifest files. The following steps constitute the pipeline. First, we extend an open source tool called Googler [18] to harvest web links that contain video content. Second, we

use the harvested links to drive a Google Chrome browser that loads the link and outputs the network requests while it completes the page download. We do this by using the chrome-remote-interface [7] which allows us to remotely interact with a Google Chrome instance. Third, we filter and collect the manifest URLs from the network log captured by Chrome. Finally, we use custom scripts to download and process the manifest files (and any sub-manifest files referenced in the main manifest file) to record data of interest along the dimensions described above. We now describe each of these steps in more detail and concretely define our extensions to the different open source tools we use.

Googler. Googler [18] is a tool written in Python that enables us to perform google searches with custom queries. Googler has support for regular google searches and searches targeting the "News" tab of the results. Additionally it can navigate over the search results (with a starting point and a total count of urls) and return them in either plain text or json format. For our needs we have extended Googler to perform searches targeting the "Videos" tab of the results. This way we can narrow down the returned urls to the ones that contain videos in them. An example of how we are using Googler in this project is the following:

```
googler –noprompt –nocolor –json -V –start 10 –count 10 -w www.cnn.com ""
```

Chrome-Remote-Interface. Chrome-Remote-Interface [7] is an interface for the Chrome Debugging Protocol that helps instrument Google Chrome and access its monitoring tools, such as console or networking, via Javascript code. For our needs we are utilizing this tool to interact with a Chrome browser instance, access the web pages that contain video and log the network requests while the page loads. Since we are mainly interested in obtaining the manifest files we do not need to wait for the video to finish. The manifest file is downloaded before the video starts to play. So we set a timeout of 30 seconds, a long enough value for the manifest file to download. After the 30 seconds we terminate the browser to free memory and continue the manifest file retrieval for the next video.

Chrome. Chrome is a web browser developed by Google. We are using Google Chrome v.65 as a standard way of accessing urls containing video, in order to hide our crawler behind a common web browser, interpret javascript at runtime (sometimes it's required for the playback) and start the video playback automatically. Also on the instance we are running we have installed the AdBlock extension v.3.27.0, which is really useful in avoiding advertisements where possible.

Custom Crawler. Our crawler is responsible for combining all these individual components into a single pipeline. It takes as input a text file that contains the websites to be searched for pages that contain video, along with starting points for the search and the total number of urls that are going to be accessed. Currently the steps that the crawler takes to retrieve the manifest files are the following:

1. Retrieve X number of urls for each website in the input file
2. Startup a Google Chrome instance in remote debug mode
3. For each url collected, interact with Google Chrome using chrome-remote-interface, access the url and retrieve the network requests
4. After the timeout period (currently 30 seconds) search the network requests for video manifest files based on known extensions (".m3u8", ".mpd", ".f4m")
5. Retrieve and store the contents of the manifest files with a simple HTTP GET

It's important to state that we have implemented a simple logging and checkpointing logic. From our crawler's execution we log the urls fetched with googler, the accessed urls with their generated network requests, and Google Chrome's debug output. With this approach we can request to get more urls and access them without starting over by simply increasing the number of urls we want to access. This can be achieved because the crawler is aware of the working directory and checks the logs and created files before performing an action.

4. EXAMPLE MANIFEST FILE

Figure 1 presents an example of an HLS master manifest file collected from a video delivered by CNN.COM, and Figure 2 presents a portion from one of its sub-manifest files. The master and the sub manifest files contain a wealth of information describing the publisher's video delivery strategy for the given video. By parsing and analyzing its content we can derive information such as:

- The CDN that hosts the manifest files. In this case the CDN is Akamai, as evident by the fully qualified domain name (`https://cnnios-f.akamaihd.net`) in the URL. If the URL doesn't directly provide information about the origin we can perform a `whois` lookup to determine the owner the domain.
- The number of alternative video bitrates offered for the specific video, by counting the unique index files. In this case seven different bitrates are offered.
- The exact bitrate levels (`BANDWIDTH` tag), codecs and video resolution used for each alternative stream, by interpreting the "preamble" of each index file.

On the other hand the sub-manifest file contains the actual chunks streamed to the device for the video playback. From this file we can get information about:

- The CDN that hosts the actual chunks of the video streamed

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=621000,RESOLUTION=640x360,CODECS="avc1.66.30, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_1_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=434000,RESOLUTION=400x224,CODECS="avc1.66.30, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_0_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=807000,RESOLUTION=640x360,CODECS="avc1.66.30, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_2_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1193000,RESOLUTION=640x360,CODECS="avc1.77.30, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_3_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2887000,RESOLUTION=1280x720,CODECS="avc1.77.30, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_4_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=5229000,RESOLUTION=1920x1080,CODECS="avc1.640028, mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_5_av.m3u8?null=0

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=38000,CODECS="mp4a.40.2",CLOSED-CAPTIONS=NONE
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/index_0_a.m3u8?null=0
```

Figure 1—Master Manifest File Example

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-ALLOW-CACHE:YES
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10.000,
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/segment1_0_a.ts?null=0
#EXTINF:10.000,
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/segment2_0_a.ts?null=0
#EXTINF:10.000,
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/segment3_0_a.ts?null=0
#EXTINF:10.000,
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/segment4_0_a.ts?null=0
#EXTINF:10.000,
https://cnnios-f.akamaihd.net/i/cnn/big/cnn10/2017/11/14/caption/ten-1115.cnn_1738140_ios_,440,650,840,1240,3000,5500,.mp4.csmil/segment5_0_a.ts?null=0
.
.
.
#EXT-X-ENDLIST
```

Figure 2—Sub Manifest File Example

- The type of video that is being viewed. In this case the video is pre-recorded on demand video (VOD) as given by the `EXT-X-PLAYLIST-TYPE` tag. Alternatively it could have been a live stream.
- The time segment of the video each chunk covers, by parsing the preamble of the file. Specifically the `EXT-X-TARGETDURATION` tag provides us information about the duration of the chunk. In this case the chunk duration is 10 seconds.

5. RESEARCH PROJECT C (DELIVERABLES)

1. Continue regular meetings with my mentor and collaboration over GitHub and Google Docs
2. Improvements on our custom crawler, such as adding multithreading support for accessing multiple urls concurrently by opening multiple browser tabs
3. Collect a representative sample of video manifest files from each publisher in order to derive accurate statistics
4. Implement analysis scripts to parse the collected manifest files and produce useful statistics and insights for the video delivery strategies
5. Summarize the findings of the study and create an informative poster for the poster session that will be held by the end of the semester
6. Make publicly available our crawler implementation and analysis scripts used in this project

6. RELATED WORK

Due to their popularity, video serving systems have been the subject of a large body of work. YouTube alone has been the subject of numerous studies over the years. [3, 6, 13, 23, 8, 12, 20, 10, 22]. Together these works have studied a number of YouTube's aspect, including the architecture and its evolution, serving strategy, characterization of videos and the user access patterns etc. Netflix, another major provider has also received much attention, where researchers have characterized its delivery strategy in [2, 14].

In addition to these well known systems, researchers have also focused on a on-demand TV publisher [1, 16], as well as novel video delivery systems which allow users to broadcast live streams [21, 19]. The thread that unites these works is their specific focus on one or a handful of online publishers, in contrast this project focuses on understanding the diversity at a macro level across a large number of online video publishers. One piece of work that is somewhat closer to this project is [11]. This work analyzes video traffic generated from a large number of users of a prominent cellular services provider. Their results show that in 2011, HLS, a video streaming protocol contributed to one third of the total video traffic.

Bibliography

[1] Henrik Abrahamsson and Mattias Nordmark. "Program Popularity and Viewer Behaviour in a Large TV-on-demand System." In: *Proceedings of the 2012 Internet Measurement Conference*. IMC '12. Boston, Massachusetts, USA, 2012 (Cited on page 3).

[2] Vijay K. Adhikari, Yang Guo, Fang Hao, Volker Hilt, Zhi-Li Zhang, Matteo Varvello, and Moritz Steiner. "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs." In: *IEEE/ACM Trans. Netw.* 23.6 (Dec. 2015) (Cited on page 3).

[3] Vijay Kumar Adhikari, Sourabh Jain, and Zhi-Li Zhang. "YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective." In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. Melbourne, Australia, 2010. ISBN: 978-1-4503-0483-2 (Cited on page 3).

[4] Adobe. *Adobe HTTP Dynamic Streaming*. www.adobe.com/products/hds-dynamic-streaming.html. (Cited on page 1).

[5] Apple. *Apple's HTTP Live Streaming*. https://developer.apple.com/streaming/. (Cited on page 1).

[6] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. "Mapping the Expansion of Google's Serving Infrastructure." In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain, 2013 (Cited on page 3).

[7] Andrea Cardaci, Andrey Sidorov, and Greg Cochard. *Chrome-Remote-Interface*. https://github.com/cyrus-and/chrome-remote-interface (Cited on page 2).

[8] 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: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. IMC '07. San Diego, California, USA, 2007. ISBN: 978-1-59593-908-1 (Cited on page 3).

[9] DASH-IF. *Dash.js*. http://dashif.org/reference/players/javascript/1.4.0/samples/dash-if-reference-player/ (Cited on page 1).

[10] Yuan Ding, Yuan Du, Yingkai Hu, Zhengye Liu, Luqin Wang, Keith Ross, and Anindya Ghose. "Broadcast Yourself: Understanding YouTube Uploaders." In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. 2011 (Cited on page 3).

[11] Jeffrey Erman, Alexandre Gerber, K. K. Ramadrishnan, Subhabrata Sen, and Oliver Spatscheck. "Over the Top Video: The Gorilla in Cellular Networks." In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. Berlin, Germany, 2011 (Cited on page 3).

[12] Alessandro Finamore, Marco Mellia, Maurizio M. Munafò, Ruben Torres, and Sanjay G. Rao. "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience." In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. Berlin, Germany, 2011 (Cited on page 3).

[13] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. "Youtube Traffic Characterization: A View from the Edge." In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. IMC '07. San Diego, California, USA, 2007. ISBN: 978-1-59593-908-1 (Cited on page 3).

[14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service." In: *Proceedings of the 2014 ACM Conference on SIGCOMM*. SIGCOMM '14. Chicago, Illinois, USA, 2014 (Cited on page 3).

[15] Alexa Internet Inc. *Alexa Top 500*. https://www.alexa.com/topsites (Cited on page 1).

[16] Zhenyu Li, Jiali Lin, Marc-Ismael Akodjenou, Gaogang Xie, Mohamed Ali Kaafar, Yun Jin, and Gang Peng. "Watching Videos from Everywhere: A Study of the PPTV Mobile VoD System." In: *Proceedings of the 2012 Internet Measurement Conference*. IMC '12. Boston, Massachusetts, USA, 2012 (Cited on page 3).

[17] Microsoft. *Microsoft Smooth Streaming*. http://www.iis.net/downloads/microsoft/smooth-streaming (Cited on page 1).

[18] Arun Prakash Jana, Henri Hakkinen, Zhiming Wang, Johnathan Jenkins, and SZ Lin. *Googler*. https://github.com/jarun/googler (Cited on pages 1 and 2).

[19] Matti Siekkinen, Enrico Masala, and Teemu Kämäräinen. "A First Look at Quality of Mobile Live Streaming Experience: The Case of Periscope." In: *Proceedings of the 2016 Internet Measurement Conference*. IMC '16. Santa Monica, California, USA, 2016 (Cited on page 3).

[20] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M. Munafo, and Sanjay Rao. "Dissecting Video Server Selection Strategies in the YouTube CDN." In: *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*. ICDCS '11. 2011 (Cited on page 3).

[21] Bolun Wang, Xinyi Zhang, Gang Wang, Haitao Zheng, and Ben Y. Zhao. "Anatomy of a Personalized Livestreaming System." In: *Proceedings of the 2016 Internet Measurement Conference*. IMC '16. Santa Monica, California, USA, 2016 (Cited on page 3).

[22] Jia Zhou, Yanhua Li, Vijay Kumar Adhikari, and Zhi-Li Zhang. "Counting YouTube Videos via Random Prefix Sampling." In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. Berlin, Germany, 2011. ISBN: 978-1-4503-1013-0 (Cited on page 3).

[23] Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose. “Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications.” In: *Comput. Netw.* (Mar. 2009) (Cited on page 3).