Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery

Vijay Kumar Adhikari*, Yang Guo†, Fang Hao†, Matteo Varvello†, Volker Hilt†, Moritz Steiner† and Zhi-Li Zhang*
*University of Minnesota, †Bell-Labs/Alcatel-Lucent

***Abstract*—Netflix is the leading provider of on-demand Internet video streaming in the US and Canada, accounting for 29.7% of the peak downstream traffic in US. Understanding the Netflix architecture and its performance can shed light on how to best optimize its design as well as on the design of similar on-demand streaming services. In this paper, we perform a measurement study of Netflix to uncover its architecture and service strategy. We find that Netflix employs a blend of data centers and Content Delivery Networks (CDNs) for content distribution. We also perform active measurements of the three CDNs employed by Netflix to quantify the video delivery bandwidth available to users across the US. Finally, as improvements to Netflix's current CDN assignment strategy, we propose a measurement-based adaptive CDN selection strategy and a multiple-CDN-based video delivery strategy, and demonstrate their potentials in significantly increasing user's average bandwidth.**

I. Introduction

Netflix is the leading subscription service provider for online movies and TV shows. Netflix attracts more than 23 million subscribers in the United States and Canada, and can stream out HD (High Definition) quality video with average bitrate reaching 3.6 Mbps. In fact, Netflix is the single largest source of Internet traffic in the US, consuming 29.7% of peak downstream traffic [1]. Its design and traffic management decisions have a substantial impact on the network infrastructure.

Designing such a large scale, fast growing video streaming platform with high availability and scalability is technically challenging. The majority of functions used to be hosted in Netflix's own data center. Recently, Netflix has resorted to the use of cloud services [2], Content Distribution Networks (CDNs), and other public computing services. Amazon AWS cloud replace in-house IT, and Amazon SimpleDB, S3 and Cassandra are used for file storage [2]. Video streaming is served out of multiple CDNs, and *UltraDNS*, a public DNS service, is used as its authoritative DNS servers. Microsoft Silverlight [3] is employed as the video playback platform for Netflix desktop users. The end result is amazing: Netflix manages to build its Internet video delivery service with little infrastructure of its own!

In this paper we provide a detailed analysis of the Netflix architecture, which is designed to serve massive amounts of content by combining multiple third party services. This architecture can be considered as a possible blue print for a scalable, infrastructure-less content provider. We discuss the interaction between the components of this design including multiple CDNs and HTTP adaptive streaming, and analyze the algorithms used by Netflix that provide the glue to piece together the overall system. We also shed light on the implications the Netflix design decisions have on CDNs, the network and the end user experience both to understand its performance and to improve its design. In addition, based on our measurement results, we suggest new video delivery strategies that can further improve user experience by effectively utilizing multiple CDNs.

Despite the popularity of Netflix, surprisingly there have been very few studies looking into its streaming service platform. The authors of [4] investigate the Netflix security framework, while the authors of [5] focus on the rate-adaptation mechanisms employed by Silverlight player and experimentally evaluated the Netflix players. To the best of our knowledge, we are the first to take a systematic look into the architecture of the Netflix video streaming together with an extensive measurement study of three CDNs it employs.

The main contributions of this paper can be summarized as follows:

- We dissect the basic architecture of the Netflix video streaming platform by monitoring the communications between the Netflix player and various components of the Netflix platform. We collect a large number of Netflix video streaming manifest files to analyze how geographic locations, client capabilities, and content type affect the streaming parameters used by Netflix, such as content formats, video quality levels, CDN ranking, and so forth.
- We analyze how Netflix makes use of multiple CDNs under changing bandwidth conditions. We find that Netflix players stay attached to a fixed CDN even when the other CDNs can offer better video quality.
- We perform an extensive bandwidth measurement study of the three CDNs used by Netflix. The results show that there is significant variation in CDN performance across time and location.
- Finally, we explore alternative strategies for improving video delivery performance by using multiple CDNs. Our study shows that selecting the best serving CDN based on a small number of measurements at the beginning of each video session can deliver more than 12% bandwidth

This study was done while Vijay Kumar Adhikari was a summer intern at Bell Labs, Alcatel-Lucent. It was supported in part by the NSF grants 08-31734, CNS-0905037, CNS-1017092 and CNS-1017647.

TABLE I
KEY NETFLIX HOSTNAMES

Hostname	Organization
www.netflix.com	Netflix
signup.netflix.com	Amazon
movies.netflix.com	Amazon
agmoviecontrol.netflix.com	Amazon
nflx.i.87f50a04.x.lcdn.nflximg.com	Level 3
netflix-753.vo.llnwd.net	Limelight
netflix753.as.nflximg.com.edgesuite.net	Akamai

improvement over the static CDN assignment strategy currently employed by Netflix. Furthermore, using multiple CDNs simultaneously can achieve more than 50% improvement.

The paper is organized as follows. Section II describes the architecture of Netflix video streaming system and CDN selection strategy. Section III presents our measurement study of the three CDNs. Section IV explores the alternative strategies for CDN assignment in order to improve video delivery performance. Section V discusses the related work. Finally, Section VI concludes the paper and discusses the future work.

II. NETFLIX VIDEO STREAMING PLATFORM

We start the section with the overview of Netflix video streaming platform architecture. We dissect the architecture via traffic monitoring, DNS resolutions, and WHOIS[6] lookup. We then present the timeline of serving a single Netflix client as an example to illustrate the interplay between a Netflix player and various service components. We further collect a large number of video streaming manifest files using *Tamper Data* add-on[7], and analyze how geographic locations, client capabilities, and content types influence the streaming parameters. Finally, we focus on the Netflix CDN assignment strategy. Using *dummynet* [8] to strategically throttle individual CDN's bandwidth, we discover how Netflix makes use of multiple CDNs in face of bandwidth fluctuation.

A. Overview of Netflix architecture

To observe the basic service behavior, we create a new user account, login into the Netflix website and play a movie. We monitor the traffic during all of this activity and record the hostnames of the servers involved in the process. We then perform DNS resolutions to collect the canonical names (CNAMEs) and IP addresses of all the server names that the browser have contacted. We also perform WHOIS[6] lookups for the IP addresses to find out their owners. Table I summarizes the most relevant hostnames and their owners. Fig. 1 shows the basic architecture for Netflix video streaming platform. It consists of four key components: Netflix data center, Amazon cloud, CDNs and players.

• *Netflix data centers*. Our analysis reveals that Netflix uses its own IP address space for the hostname `www.netflix.com`. This server primarily handles two key functions: (a) registration of new user accounts and capture of payment information (credit card or Paypal account), and (b) redirect users to `movies.netflix.com` or

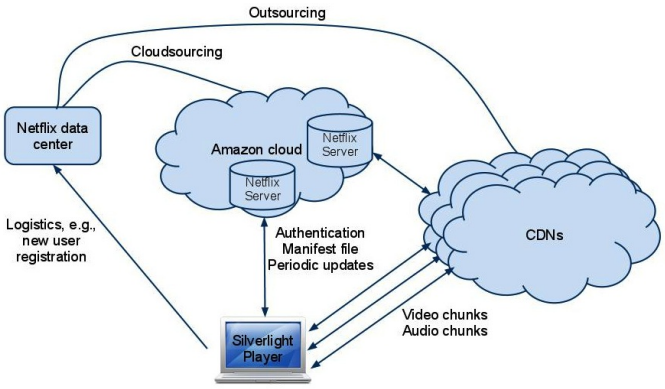

Fig. 1. Netflix architecture

`signup.netflix.com` based on whether the user is logged in or not respectively. This server does not interact with the client during the movie playback, which is consistent with the recent presentation from Netflix team [9].

• *Amazon cloud*. Except for `www.netflix.com` which is hosted by Netflix, most of the other Netflix servers such as `agmoviecontrol.netflix.com` and `movies.netflix.com` are served off the Amazon cloud [10]. [9] indicates that Netflix uses various Amazon cloud services, ranging from EC2 and S3, to SDB and VPC [10]. Key functions, such as content ingestion, log recording/analysis, DRM, CDN routing, user sign-in, and mobile device support, are all done in Amazon cloud.

• *Content Distribution Networks (CDNs)*. Netflix employs multiple CDNs to deliver the video content to end users. The encoded and DRM protected videos are sourced in Amazon cloud and copied to CDNs. Netflix employs three CDNs: *Akamai*, *LimeLight*, and *Level-3*. For the same video with the same quality level, the same encoded content is delivered from all three CDNs. In Section II-D we study the Netflix strategy used to select these CDNs to serve videos.

• *Players*. Netflix uses Silverlight to download, decode and play Netflix movies on desktop web browsers. The run-time environment for Silverlight is available as a plug-in for most web browsers. There are also players for mobile phones and other devices such as Wii, Roku, etc. This paper, however, focuses on Silverlight player running on desktop PCs.

Netflix uses the DASH (*Dynamic Streaming over HTTP*) protocol for streaming. In DASH, each video is encoded at several different quality levels, and is divided into small 'chunks' - video segments of no more than a few seconds in length. The client requests one video chunk at a time via HTTP. With each download, it measures the received bandwidth and runs a *rate determination algorithm* to determine the quality of the next chunk to request. DASH allows the player to freely switch between different quality levels at the chunk boundaries.

B. Servicing a Netflix client

We now take a closer look at the interaction between the client web browser and various web servers involved in the

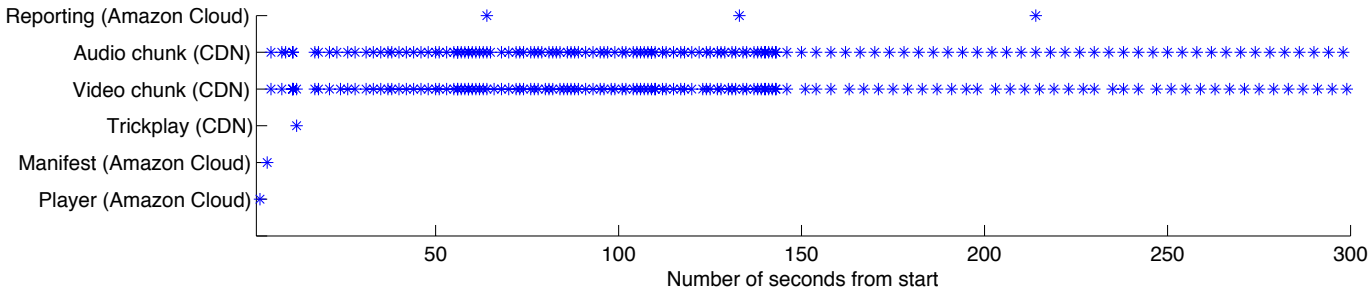

Fig. 2. Timeline in serving a Netflix client

video playback process. Fig. 2 shows the timeline along which the streaming service is provided to a desktop client, and indicates the involved server entities. The X-axis in this figure shows the time from the beginning of the experiment to 5 minutes and the Y-axis lists different activities. The client first downloads the Microsoft Silverlight application from `movies.netflix.com` and authenticates the user. After authentication, the player fetches the manifest file from the control server at `agmoviecontrol.netflix.com`, based on which it starts to download trickplay data and audio/video chunks from different CDNs. Client reports are sent back to the control server periodically. We describe further details of individual activities below.

1) Silverlight player download and user authentication: Video playback on a desktop computer requires the Microsoft Silverlight browser plug-in to be installed on the computer. When the user clicks on "Play Now" button, the browser downloads the Silverlight application and then that application starts downloading and playing the video content. This small Silverlight application is downloaded for each video playback.

2) Netflix manifest file: Netflix video streaming are controlled by instructions in a manifest file that the Silverlight client downloads. The Netflix manifest file provides the DASH player metadata to conduct the adaptive video streaming. The manifest files are client-specific, i.e., they are generated according to each client's playback capability. For instance, if the user player indicates it is capable of rendering h.264 encoded video, h.264 format video is included in the manifest file. If the player indicates that it can only play back .wmv format, only .wmv format video is included.

The manifest file is delivered to end user via SSL connection and hence the content of the file cannot be read over the wire using packet capture tools such as *tcpdump* or *wireshark*. We use Firefox browser and *Tamper Data* plug-in to extract the manifest files. The extracted manifest file is in XML format and contains several key pieces of information including the list of the CDNs, location of the trickplay data, video/audio chunk URLs for multiple quality levels, and timing parameters such as time-out interval, polling interval and so on. The manifest file also reveals interesting information on the Netflix system architecture. For instance, they show that Netflix uses three CDNs to serve the videos. Different ranks are assigned to different CDNs to indicate to the clients which CDN is more preferred than others. A section of one of the manifest files is shown in Fig. 3, where Level3 is listed as the most preferred CDN for this client. We will conduct more elaborate experiments and discuss more details of the manifest files later in this section.

```
<nccp:cdns>
    <nccp:cdn>
        <nccp:name>level3</nccp:name>
        <nccp:cdnid>6</nccp:cdnid>
        <nccp:rank>1</nccp:rank>
        <nccp:weight>140</nccp:weight>
    </nccp:cdn>
    <nccp:cdn>
        <nccp:name>limelight</nccp:name>
        <nccp:cdnid>4</nccp:cdnid>
        <nccp:rank>2</nccp:rank>
        <nccp:weight>120</nccp:weight>
    </nccp:cdn>
    <nccp:cdn>
        <nccp:name>akamai</nccp:name>
        <nccp:cdnid>9</nccp:cdnid>
        <nccp:rank>3</nccp:rank>
        <nccp:weight>100</nccp:weight>
    </nccp:cdn>
</nccp:cdns>
```

Fig. 3. CDN list in manifest file

3) Trickplay: Netflix Silverlight player supports simple trickplay such as pause, rewind, forward and random seek. Trickplay is achieved by downloading a set of thumbnail images for periodic snapshots. The thumbnail resolution, pixel aspect, trickplay interval, and CDN from where to download the trickplay file are described in the manifest file. The trickplay interval for the desktop browser is 10 seconds, and multiple resolutions and pixel aspects for trickplay are provided.

4) Audio and video chunk downloading: As shown in Fig. 2, audio and video contents are downloaded in chunks. Download sessions are more frequent at the beginning so as to build up the player buffer. Once the buffer is sufficiently filled, downloads become periodic. The interval between the beginning of two consecutive downloads is approximately four seconds - the playback length of a typical chunk.

The manifest file contains multiple audio and video quality levels. For each quality level, it contains the URLs for individual CDNs, as shown in Fig. 4.

```
<nccp:bitrate>560</nccp:bitrate>
<nccp:videoprofile>
   playready-h264mpl30-dash
</nccp:videoprofile>
<nccp:resolution>
   <nccp:width>512</nccp:width>
   <nccp:height>384</nccp:height>
</nccp:resolution>
<nccp:pixelaspect>
   <nccp:width>4</nccp:width>
   <nccp:height>3</nccp:height>
</nccp:pixelaspect>v
<nccp:downloadurls>
   <nccp:downloadurl>
      <nccp:expiration>131xxx</nccp:expiration>
      <nccp:cdnid>6</nccp:cdnid>
      <nccp:url>http://nflx.i.../...</nccp:url>
   </nccp:downloadurl>
   <nccp:downloadurl>
      <nccp:expiration>131xxx</nccp:expiration>
      <nccp:cdnid>4</nccp:cdnid>
      <nccp:url>http://netflix.../...</nccp:url>
   </nccp:downloadurl>
   <nccp:downloadurl>
      <nccp:expiration>131xxx</nccp:expiration>
      <nccp:cdnid>9</nccp:cdnid>
      <nccp:url>http://netflix.../...</nccp:url>
   </nccp:downloadurl>
</nccp:downloadurls>
```

Fig. 4. Video downloadable for one quality level

5) User experience reporting: After the playback starts, Netflix player communicates periodically with the control server `agmoviecontrol.netflix.com`. Based upon the keywords such as "/heartbeat" and "/logblob" in the request URLs and the periodicity of the communication, we conjecture that they are periodic keep alive messages and log updates. However, the actual messages that we have extracted by using *Tamper Data* do not appear to be in clear text and hence we cannot verify it further.

C. Manifest file analysis

A manifest file is delivered over the SSL connection. We use *Tamper Data* plug-in for Firefox browser to read the file. Since the manifest files contain a wealth of information and shed lights on the Netflix strategies, we conduct a large scale experiment by collecting and analyzing a number of manifest files. We are interested in understanding how geographic locations, client capabilities, and content type (e.g., popular vs unpopular, movies vs TV shows) may impact the streaming parameters. We use six different user accounts, 25 movies of varying popularity, age and type, four computers with Mac and Windows systems at four different locations for this experiment. From each computer, we log into Netflix site using each of the user accounts and play all of the movies for few minutes to collect the manifest files. In addition to using client machines located in different geographies, we also configure those client browsers to use Squid proxy servers running on ten PlanetLab nodes hosted by US universities in different geographic regions to collect additional manifest files.

1) CDN ranking and user accounts: Netflix manifest files rank CDNs to indicate which CDNs are preferred. CDN ranking determines from which CDN the client downloads the video and may affect user perceived video quality. We analyze the collected manifest files to understand the factors that affect the rankings of the CDNs. For this analysis, we build a table that lists CDN ranking for each combination of user account, client computer (or PlanetLab proxy), movie ID and time of day for several days. Analysis of this table suggests that the CDN ranking is only based upon the user account. For a given user account, the CDN ranking in the manifest file remains the same irrespective of movie types, computers, time and locations. Furthermore, for the same movie, computer, location and around the same time, two different users may see different CDN rankings. We also observe that the CDN ranking for each user account remains unchanged for at least several days. As we show in measurement results in the next section, such assignment of ranking seems to be independent of available bandwidth from each CDN.

2) Audio/Video bit rates: Netflix serves videos in multiple formats and bitrates. When a Netflix client requests for the manifest file from Netflix, the client indicates the formats of the content it can play. Netflix server then sends back a manifest file based upon the client request. For instance, Netflix client running on an older computer (Thinkpad T60 with Windows XP) and a newer computer (Macbook Pro with Snow Leopard) have different capabilities and receive different video downloading format and bit rates.

Based on the client capabilities, the server sends URLs for the video and audio chunks in the returned manifest files. In general, manifest files contain information about video chunks encoded in bitrates between 100Kbps to 1750Kbps (and 2350Kbps and 3600Kbps for videos available in HD) for the manifest files sent to the newer computer. We see that videos available in HD can be served in up to 14 different bitrates whereas non-HD content can be served in up to 12 different bitrates. We also note that Netflix clients do not try all possible available bitrates when trying to determine the optimal playback rate.

D. CDN selection strategy

We have seen that a Netflix client can choose different video bitrates and different CDNs for video downloading. In this section we conduct experiments to help understand how Netflix make such choices when bandwidth is dynamic. We play a single movie from the beginning. Once the playback starts, we gradually throttle the available bandwidth of the top ranked CDN in the manifest file. We use *dummynet* to throttle the inbound bandwidth to the client.

At the beginning, servers from each CDN are allowed to send data at 3,900Kbps. After every minute, we reduce the available bandwidth for the current CDN by 100Kbps till it reaches 100Kbps. At that point we start throttling the next CDN in the same way and so on. We plot our observation in Fig. 5. In this figure, the X-axis shows the time starting from the beginning of playback. The Y-axis shows both the throttled bandwidth and the playback rate. In this instance, Level3, Limelight and Akamai CDNs are ranked first, second

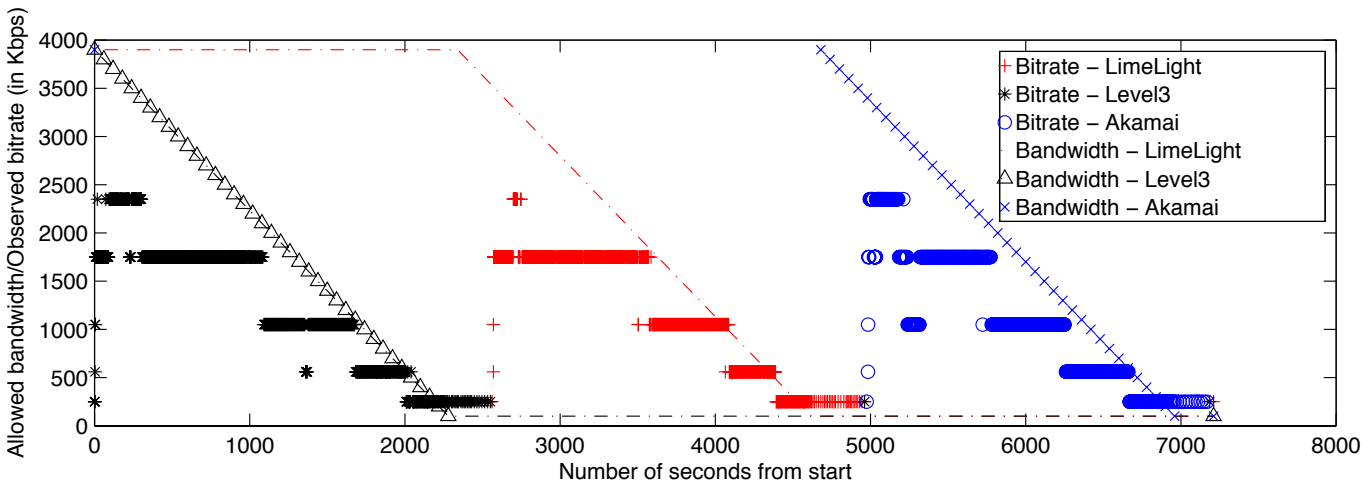

Fig. 5. CDN switching

Fig. 6. Best CDN at each vantage point

and third respectively. The client starts downloading video chunks from the first CDN. In the beginning, it starts from a low bitrate and gradually improves the bitrate in a probing fashion. As we lower the available bandwidth for the first CDN while leaving the other CDNs intact, we notice something interesting. Instead of switching to a different CDN, which is not throttled, the client keeps lowering the bitrate and stays with the first CDN. Only when it can no longer support even the very low quality level (i.e. when the available bandwidth for the first CDN reaches 100Kbps), it switches to the second CDN. It repeats almost the same behavior as we leave the first CDN at 100Kbps and gradually lower the available bandwidth for the second CDN while leaving the third CDN intact. In general, the Netflix client appears to stay with the same CDN as long as possible even if it has to degrade the quality level of the playback.

III. CDN Performance Measurement

We have observed in the previous section that Netflix CDN ranking is tied to each user account and remains unchanged over many days. Even a change in the user geographic location does not trigger any CDN ranking change. Such assignment strategy naturally leads to the following questions:

- How does each CDN perform? Can the selected CDN server consistently support the bandwidth needed for high quality streaming?
- How do different CDNs compare in terms of performance? Is any CDN clearly better or worse than others?
- How far is the current Netflix assignment strategy from "optimal"?
- Is it possible to improve the assignment strategy to support higher delivery bandwidth?

In the next two sections, we attempt to address the above questions by conducting extensive measurement experiments for the three CDNs used by Netflix from 95 vantage points across the United States.

We measure the bandwidth throughput between each vantage point and a given CDN server by downloading multiple video chunks from the CDN server. Video file URLs are collected for all three CDNs from manifest files. Here we take advantage of the fact that the URLs in the manifest remain valid for several hours from the time the manifest file is generated, and the validity of the URLs are not tied to client IP address. Furthermore, the byte "range" of the download can be adjusted without affecting the URL validity. Once we extract the URLs for the three CDNs, we "replay" the GET request from all vantage points with byte range modified so that we download video chunks of the same size.

Similar to the actual Netflix video playback, when GET requests are sent from a vantage point, the hostnames in the URLs are resolved by DNS server, which returns the IP address of the edge server assigned by the CDN. To ensure the measured bandwidth of three CDNs are comparable, we send GET requests to three CDNs in round-robin order within a short duration. More specifically, measurement is repeated in multiple "rounds", with each round lasting 96 seconds. A round is further partitioned into four "slots", with 24 seconds for each slot. The first three slots of each round correspond to three CDNs, respectively, and we download video chunks of size 1.8MByte. The last slot of each round

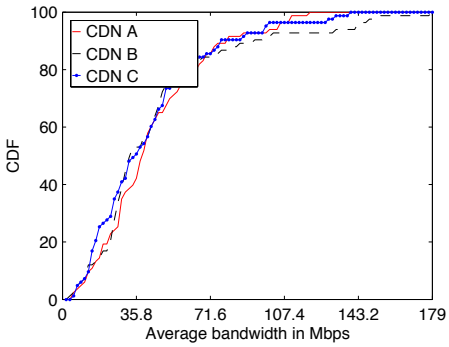

Fig. 7. CDF of average bandwidth for three CDNs measured at 83 PlanetLab nodes

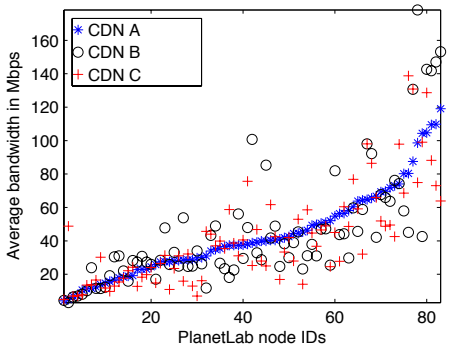

Fig. 8. Average bandwidth for the three CDNs at different PlanetLab nodes over the entire period

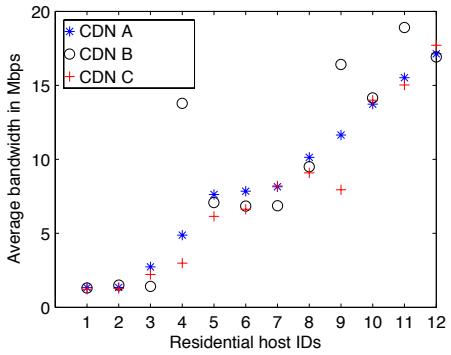

Fig. 9. Average bandwidth for the three CDNs at different residential networks over the entire period

is for a "joint" measurement for all CDNs, i.e., we send GET requests to the three CDNs simultaneously, each requesting video chunks for 0.6MByte data. We intend to find out how much total bandwidth one can get if all three CDNs are used simultaneously. We pick the size of the chunks and length of "slots" based upon multiple trial measurements. In our trials, we find that these numbers make sure that different experiments do not interfere with each other and chunk size is sufficiently large so that we can have a good estimate of the bandwidth. We also send keep-alive messages to each server every second when no data is transferred to make sure that the TCP session is alive and sender window size does not drop.

The measurement is conducted for two hours between 8 to 10pm CST, from June 8, 2011 to June 26, 2011. Based on downloading time, we calculate the instantaneous bandwidth (i.e., throughput for each GET request), the one-day average bandwidth (average bandwidth during the two hour period), and average bandwidth (over entire measurement study). These metrics allow us to examine CDN performance at multiple timescales.

We have conducted experiments from both residential sites and PlanetLab nodes. There are 12 residential sites, 10 in New Jersey, 1 in Minnesota, and 1 in California. The residential sites spread across 5 different service providers. To cover a wider range of geographic locations, we also choose 83 PlanetLab nodes spread across the United States as additional vantage points. We ensure that all selected PlanetLab nodes are lightly loaded so that the nodes themselves do not become the bottleneck and the measurement results reflect the actual bandwidth that can be supported by the CDN server and the network.

The rest of this section attempts to address the first two questions on CDN performance. We will further investigate the other two questions on performance improvement in the Section IV. We use CDN A, B, and C to denote the three CDNs without particular order in the rest of the discussion.

A. Overall CDN performance

Fig. 6 shows the locations of all vantage points in our experiments as well as the CDN with highest average bandwidth at each vantage point during the measurement period. As the result indicates, no CDN clearly outperforms the others. In addition, Fig. 7 shows the CDF (*Cumulative Distribution Function*) of average bandwidth at the PlanetLab nodes over the entire measurement period. The available bandwidth at different PlanetLab nodes varies significantly from location to location, ranging from 3Mbps to more than 200Mbps. The CDF curves of three CDNs, however, are close to each other, indicating similar overall performance.

Figures 8 and 9 further show the average bandwidth at individual locations for PlanetLab nodes and residential sites, respectively. The location index is sorted in the ascending order of CDN A's average bandwidth. CDN bandwidth measured at PlanetLab nodes appear to have much higher than that of residential sites in general. This is because most PlanetLab nodes are located in universities, which typically have better access links. This also implies that in most cases, the last mile is still the bottleneck for streaming video. However, even the residential sites with relatively low bandwidth, e.g. home 1 and 2 in Fig. 9, can support 1.3Mbps on average, enough for standard definition (SD) videos.

It is also interesting to note that home sites 4, 9, and 11 see significantly different average bandwidth from different CDNs. In particular, CDN B outperforms all others by a large margin. We find that these three homes use the same service provider. It is conceivable that CDN B has a better presence in this provider's network.

B. Daily bandwidth variation

Next we examine the bandwidth variation at different sites from different CDNs over different timescales. We compute the coefficient of variance of the daily average bandwidth at all PlanetLab nodes by computing the ratio of the standard deviation to the mean at each of the locations. Fig. 10 shows the coefficient of variance for the one-day average bandwidth at different PlanetLab nodes over multiple days. We indeed see high coefficient of variance at most nodes. The average coefficient of variance is 0.33, 0.30, and 0.30 for CDN A, B and C, respectively. At most locations, there is a significant variation in daily bandwidth for all three CDNs. We show a few representative locations in Figures 11, 12 and 13, which plot the one-day average bandwidth over the

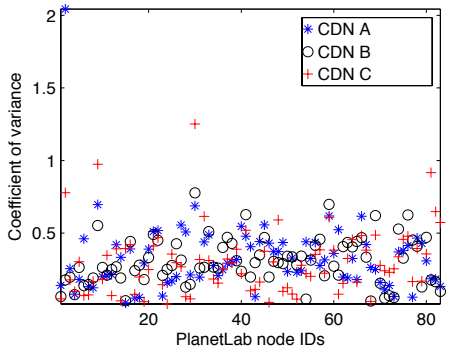

Fig. 10. Coefficient of variance for the one-day average bandwidth at different PlanetLab nodes

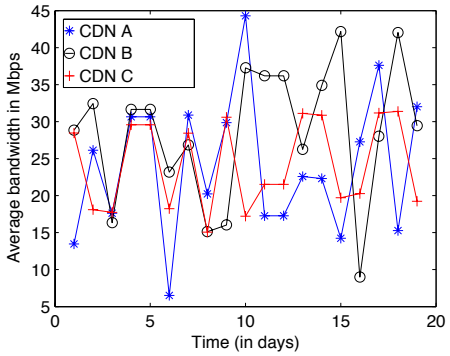

Fig. 11. One-day average bandwidth at a PlanetLab node over time

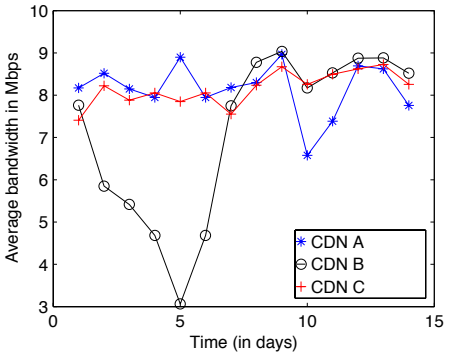

Fig. 12. One-day average bandwidth over time at residential site 7

measurement period at one PlanetLab node and two residential sites, respectively. The results show significant variation of average bandwidth on a daily basis.

Furthermore, Figures 11, 12 and 13 show that the performance ranking of different CDNs also change over time. Although the lowest CDN bandwidth across all three nodes is still above 3Mbps, sufficient to support standard definition (SD) levels, the significant variation in bandwidth and ranking of CDNs indicates a good potential to further increase bandwidth for future higher quality video delivery if better CDN selection strategy is used.

C. Variation in instantaneous bandwidth

We further investigate the instantaneous bandwidth variation during two hours of video playing. This is important since a DASH player constantly monitors the available bandwidth to decide which quality level of video to download. The small time scale bandwidth may significantly impact the Netflix users' viewing experience as two hours is a typical length of movie. Figures 14, 15 and 16 show the comparison of three CDNs for the same PlanetLab node and residential nodes. Although the variance is still significant, there is a "pattern" in the bandwidth change. For example, bandwidth for CDN B in Fig. 14 alternates between two levels, one around 35 Mbps and one around 20 Mbps. The average coefficient of variation for two hour period is 0.19, 0.21 and 0.18 respectively for CDNs A, B and C respectively for residential sites.

IV. Alternate Video Delivery Strategies

From the measurement study, we observe that Netflix statically assigns a CDN to users for extended period of time. Although all three CDNs are available, each user only uses one in most cases. Other CDNs appear to serve only as backups and are used only if current CDN server cannot support even the lowest video quality. On the other hand, our study also shows that the available bandwidth on all three CDNs vary significantly over time and over geographic locations. For instance, as shown in Fig. 6, out of 83 PlanetLab locations, CDNs A, B, and C perform best at 30, 28 and 25 locations, respectively. The measurement study of residential hosts shows similar results. If users are tied to a bad CDN choice, their video viewing quality may suffer even though other CDNs can provide them with more satisfying experience. In addition to improving experience for "unlucky" users, exploring potential ways of increasing video delivery bandwidth may also open doors for new bandwidth-demanding services in future, e.g., 3D movies or multiple concurrent movies in the same household.

In this section, we first determine how much room there is for further improvement. In other words, if we could have the optimal CDN selection strategy in theory, how much better it would be compared to current static assignment. We then explore two alternative strategies for CDN assignment that can easily be used in practice, and demonstrate we can indeed significantly increase the bandwidth for video delivery to users despite the simplicity of such strategies.

A. Room for improvement

Given the instantaneous bandwidth trace, the optimal CDN selection strategy is to choose the top CDN at each point of time. Although this cannot be done in practice since we do not know the instantaneous bandwidth beforehand, this theoretical optimal strategy allows us to find out the highest bandwidth each client can receive if the best (one) CDN is used at any given point of time. We refer to the average bandwidth achieved by the optimal strategy as the *upper bound average bandwidth*.

Fig. 17 and Fig. 18 show the average bandwidth of three CDNs and the upper bound average bandwidth for residential sites and PlanetLab nodes respectively. Here we use the average bandwidth over all three CDNs to reflect the static assignment strategy. The actual assignment may of course be better or worse depending on which CDN gets selected, but this gives the expected value. We also show the bandwidth if one top CDN, i.e., the one with highest average bandwidth is selected. For the majority of the sites, the upper bound is much better than the average CDN case, and close to the top CDN case. In particular, the upper bound is 17% and 33% better than the average case for residential sites and PlanetLab nodes respectively, indicating there is significant room for improvement. Assigning users to top CDN is only 6% to 7% worse than the theoretical optimal case. This indicates that if

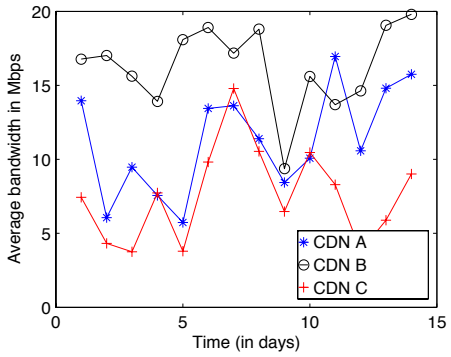

Fig. 13. One-day average bandwidth over time at residential site 9

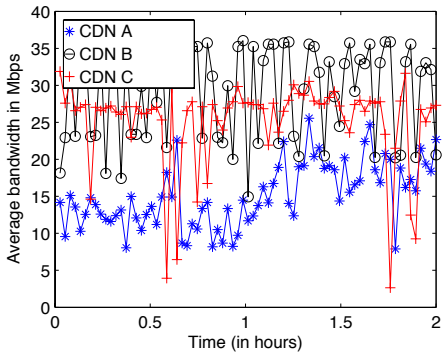

Fig. 14. Instantaneous bandwidth for the three CDNs at a PlanetLab node

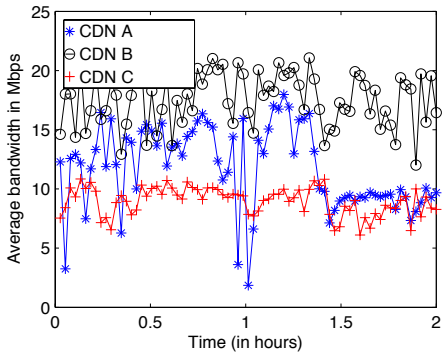

Fig. 15. Instantaneous bandwidth for the three CDNs at residential site 7

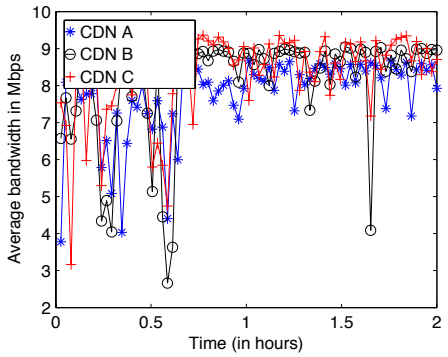

Fig. 16. Instantaneous bandwidth for the three CDNs at 9-th residential site

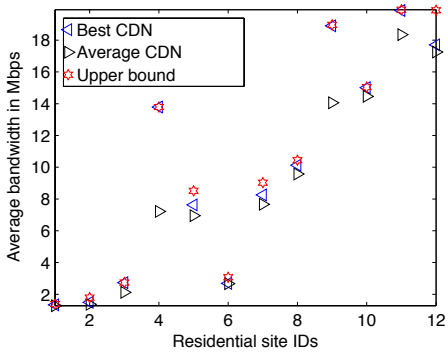

Fig. 17. Average bandwidth for three CDNs and the upper bound at residential sites

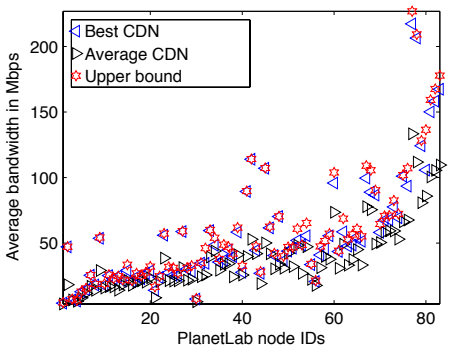

Fig. 18. Average bandwidth for three CDNs and the upper bound at PlanetLab nodes

we can estimate which CDN is likely to perform best in next couple hours, we can achieve average bandwidth that is fairly close to the *upper bound average bandwidth*.

B. Measurement based CDN selection

Since selecting the top CDN for users gives good performance, we next study how to identify the top CDN effectively. We propose to have the player conduct the instantaneous bandwidth measurement multiple times at the beginning, and assign users the best-performing CDN for the rest of the movie. Fig. 19 shows the effect of number of measurements on performance. As reference, two straight lines show the ratio of the CDN average bandwidth over top CDN bandwidth for all PlanetLab and residential nodes, respectively. In both cases we calculate the average CDN bandwidth over all locations, time, and CDN providers, so they reflect the expected CDN performance, assuming the three CDNs are equally likely to be chosen in the static CDN assignment strategy. The other two curves are ratio of average bandwidth using measurement based CDN selection strategy over that of using top CDN for both PlanetLab nodes and residential sites. Using a small number of measurements ($\geq$ 2), the measurement based strategy delivers more than 12% improvement over the static CDN assignment strategy. Although the average improvement is moderate, for certain users the improvement is significant, e.g., more than 100% for residential host 4. Given this method is very straightforward and easy to implement, we believe this is a favorable approach for improving video delivery.

C. Using multiple CDNs simultaneously

In previous sections, we have assumed that only one CDN can be used at a time. However, since Silverlight player downloads video and audio content in chunks, it is possible to use all three CDNs simultaneously. For instance, the player can download three different chunks in parallel from three different CDNs to obtain larger bandwidth. Since the design of a HTTP adaptive streaming protocol that can best utilize multiple CDNs is out of the scope of this paper, we try to see if multiple CDNs can be used, whether they can offer higher aggregated throughput for end users.

Fig. 20 and Fig. 21 compare the average bandwidth using top CDN and the average bandwidth obtained by combining three CDNs for residential and PlanetLab nodes, respectively. We see that combining all three CDNs can significantly improve the average bandwidth. Specifically, the aggregate bandwidth obtained by combining all 3 CDNs is greater than the bandwidth of the single best CDN by 54% to 70% for residential sites and PlanetLab nodes, respectively.

V. Related Work

Several recent works have been done in analyzing different aspects of Netflix video streaming. Akhshabi et al.

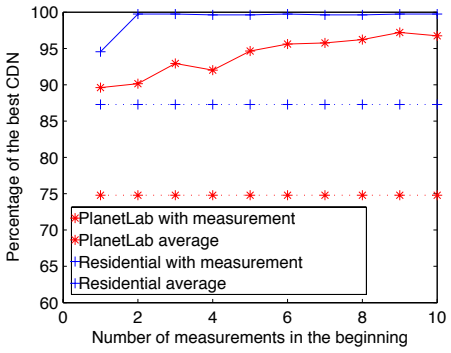

Fig. 19. Measurement based CDN selection: effect of number of measurements

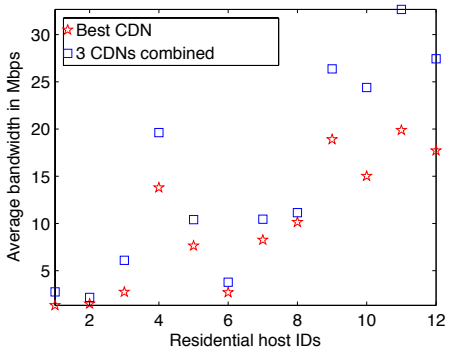

Fig. 20. Bandwidth from best CDN vs three combined CDNs for residential hosts

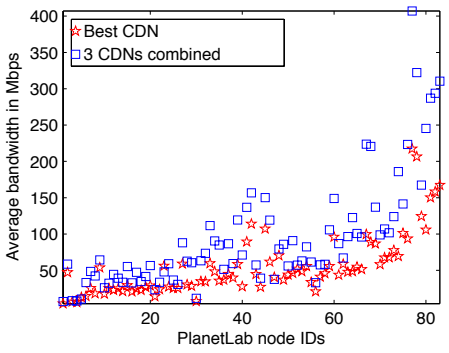

Fig. 21. Bandwidth from best CDN vs three combined CDNs for PlanetLab nodes

have studied several video streaming players including Netflix player and investigated how the streaming clients react to bandwidth changes [5]. The measurement is done mostly from one fixed location. Pomelo has also presented an interesting analysis of Netflix security framework [4]. This work was done before Netflix is migrated into Amazon cloud. Unlike the previous work, we investigate a broader set of components in Netflix video delivery system, and focus on how the player interacts with different CDNs. To achieve this, we conduct more extensive measurement from multiple geo locations.

Recent work has also been done for other streaming platforms [11], [12]. Krishnappa et al. have studied Hulu streaming with emphasis on improving performance using prefetching and caching [11]. Adhikari et al. build a measurement infrastructure by using PlanetLab nodes with the goal to understand the YouTube system architecture [12]. Unlike our work, such works do not cover behavior of multiple CDNs.

Extensive work has been done to study CDNs such as Akamai, Limelight and Google/YouTube [13], [14], [12]. But most work has been focusing on measurement of latency and does not cover the scenario where the client interacts with multiple CDNs.

Many techniques have been proposed to measure available bandwidth on a network path before, such as pathchar [15], pathload [16] and FabProbe [17]. However, they are not suitable for our study for two reasons. First, both pathchar and pathload require control at the target machine of the measurement. Second, all such tools only measure the in-path bandwidth and they cannot capture possible bandwidth shaping at the server side. Additionally, using our method more accurately reflects the download speed over HTTP than other generic methods.

VI. Conclusions

In this paper, we perform active and passive measurements to uncover the overall architecture of Netflix, currently the leading on-demand video streaming Internet provider. Since Netflix uses multiple Content Delivery Networks (CDNs) to deliver videos to its subscribers, we measure the available bandwidth of employed CDNs, and investigate its behavior at multiple time scales and at different geographic locations. We find that conducting light-weighted measurement at the beginning of the video playback and choosing the best-performing CDN can improve the average bandwidth by more than 12% than static CDN assignment strategy, and using all three CDNs simultaneously can improve the average bandwidth by more than 50%. This can be very beneficial for future bandwidth-demanding services such as 3D movies.

Future work can proceed in couple of directions. We are interested in looking into other video streaming delivery systems such as Hulu, to see if cloudsourcing and/or multiple CDN strategy have been adopted. Another venue is to develop practical HTTP adaptive streaming system that can actually utilize multiple CDNs simultaneously.

References

[1] Sandvine, "Global Internet Phenomena Report, Spring 2011," http://www.sandvine.com/news/global_broadband_trends.asp, 2011.

[2] A. Cockroft, C. Hicks, and G. Orzell, "Lessons Netflix Learned from the AWS Outage," Netflix Techblog, 2011.

[3] "Microsoft Silverlight," http://www.microsoft.com/silverlight/.

[4] Pomelo LLC, "Analysis of Netixs security framework for 'Watch Instantly' service." 2009.

[5] S. Akhshabi *et al.*, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *MMSys'11*.

[6] L. Daigle, "WHOIS Protocol Specification," RFC 3912, 2004.

[7] "Tamper Data," addons.mozilla.org/en-US/firefox/addon/tamper-data.

[8] "Dummynet," http://info.iet.unipi.it/~luigi/dummynet/.

[9] A. Cockroft, "Netflix Cloud Architecture," Velocity conference, 2011.

[10] "Amazon Web Services," http://aws.amazon.com.

[11] D. K. Krishnappa *et al.*, "On the feasibility of prefetching and caching for online tv services: a measurement study on hulu," in *PAM'11*.

[12] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, "Vivisecting youtube: An active measurement study," in *INFOCOM'12 Mini-conference*, 2012.

[13] A.-J. Su *et al.*, "Drafting behind akamai: inferring network conditions based on cdn redirections," *IEEE/ACM Trans. Netw.*, 2009.

[14] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid cdn-p2p: why limelight needs its own red swoosh," in *NOSSDAV '08*.

[15] A. Downey, "Using pathchar to estimate internet link characteristics," in *ACM SIGCOMM CCR*, 1999.

[16] M. Jain and C. Dovrolis, " Pathload: A Measurement Tool for End-to-End Available Bandwidth ," in *PAM'02*, March 2002.

[17] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. W. Biersack, "Fast available bandwidth sampling for adsl links: Rethinking the estimation for larger-scale measurements," in *PAM '09*, 2009, pp. 67–76.