Measurement Study of Netflix, Hulu, and a Tale of Three CDNs

Vijay K. Adhikari, Yang Guo, Fang Hao, *Member, IEEE*, Volker Hilt, *Member, IEEE*, Zhi-Li Zhang, *Fellow, IEEE, Member, ACM*, Matteo Varvello, and Moritz Steiner

***Abstract*—Netflix and Hulu are leading Over-the-Top (OTT) content service providers in the US and Canada. Netflix alone accounts for 29.7% of the peak downstream traffic in the US in 2011. Understanding the system architectures and performance of Netflix and Hulu can shed light on the design of such large-scale video streaming platforms, and help improving the design of future systems. In this paper, we perform extensive measurement study to uncover their architectures and service strategies. Netflix and Hulu bear many similarities. Both Netflix and Hulu video streaming platforms rely heavily on the third-party infrastructures, with Netflix migrating that majority of its functions to the Amazon cloud, while Hulu hosts its services out of Akamai. Both service providers employ the same set of three content distribution networks (CDNs) in delivering the video contents. Using active measurement study, we dissect several key aspects of OTT streaming platforms of Netflix and Hulu, e.g., employed streaming protocols, CDN selection strategy, user experience reporting, etc. We discover that both platforms assign the CDN to a video request without considering the network conditions and optimizing the user-perceived video quality. We further conduct the performance measurement studies of the three CDNs employed by Netflix and Hulu. We show that the available bandwidths on all three CDNs vary significantly over the time and over the geographic locations. We propose a measurement-based adaptive CDN selection strategy and a multiple-CDN-based video delivery strategy that can significantly increase users' average available bandwidth.**

***Index Terms*—CDN selection strategy, content distribution networks (CDN), Hulu, Netflix, Over-the-Top (OTT) content service, video streaming.**

I. Introduction

NETFLIX and Hulu are the leading subscription-based video streaming service providers for movies and TV shows. By April 2014, Netflix has attracted more than 35 million subscribers in the US alone and about 48 million worldwide [1]. It is the single largest source of Internet traffic, consuming 29.7% of peak downstream traffic in 2011 [2]. Like Netflix, Hulu also has a large viewer base, with 38 million casual viewers who watch Hulu at least once a year and 3 million paying subscribers. Both providers offer video at multiple quality levels, capable of adapting to the user's available bandwidth. Designing such large-scale, fast-growing video streaming platforms with high availability and scalability is technically challenging. Because of their popularity and size, the design and traffic management decisions of these services also have a profound impact on the Internet infrastructure.

In this paper, we provide a detailed analysis of the Netflix and Hulu architectures, which are designed to serve a massive amount of content by combining multiple third-party services. For instance, Netflix heavily utilizes the Amazon cloud service [3], replacing in-house IT by Amazon Web Service (AWS) and using Amazon SimpleDB, S3, and Cassandra for file storage [3]. Microsoft Silverlight [4] is employed as the video playback platform for Netflix desktop users. Both Netflix and Hulu use Akamai, Limelight, and Level3 content distribution networks (CDNs) for video content delivery. Such third-party service-based architecture can be regarded as a system design blueprint by future Over-the-Top (OTT) content service providers.

Despite the popularity of Netflix and Hulu, surprisingly few studies have been looking into their streaming service platforms. In order to understand the *architectural design issues* of such large-scale video streaming service platforms and their *implications*, we conducted extensive measurement studies from June to October 2011, with initial results being published in [5] and [6]. This present paper integrates/unifies the results from [5] and [6] and offers a comprehensive treatment of the two most popular content distribution platforms. The issues common for both platforms are carefully compared, while their differences are stressed. In particular, we have investigated the interactions between different components of such an architecture and analyzed the strategies used by Netflix and Hulu that provide the glue to piece together the overall system. We have also looked into the implications of these design decisions on CDNs, underlying ISP networks, as well as end-user quality of experience (QoE).

To the best of our knowledge, we are the first to take a systematic look into the architecture of these video streaming platforms together with an extensive measurement study of three CDNs they employ. Our results suggest the plausible role of *business relations* between Netflix/Hulu and CDNs in constraining how a content provider decides which CDN to select to serve streaming videos to end-users, and reveal the differing CDN selection strategies used by Netflix and Hulu to meet the business constraints. Furthermore, our measurement

Manuscript received November 14, 2013; revised May 27, 2014; accepted August 07, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Mahajan. Date of publication October 02, 2014; date of current version December 15, 2015. The work of V. K. Adhikari and Z.-L. Zhang was supported in part by the US NSF under Grant CNS-1017647 and CNS-1117536, the DTRA under Grant HDTRA1-09-1-0050, and the DoD ARO MURI under Award W911NF-12-1-0385.

V. K. Adhikari is with Microsoft, Redmond, WA 98052 USA.

Y. Guo, F. Hao, V. Hilt, and M. Varvello are with Bell Labs, Alcatel-Lucent, Holmdel, NJ 07733 USA (e-mail: yang.guo@alcatel-lucent.com).

Z.-L. Zhang is with the University of Minnesota, Twin Cities, Minneapolis, MN 55416 USA.

M. Steiner was with Bell Labs, Alcatel-Lucent, Holmdel, NJ 07733 USA. He is now with Akamai Technologies, San Francisco, CA 94103 USA.

Digital Object Identifier 10.1109/TNET.2014.2354262

results demonstrate that the CDN selection strategies employed by Netflix and Hulu do not *necessarily* provide the *best possible* QoE to end-users, thus highlighting a key *tradeoff* in CDN selection decision making: *business constraints versus end-user QoE*. To illustrate how this tradeoff can be effectively exploited, we propose new video delivery strategies that can significantly improve the user QoE by effectively utilizing multiple CDNs while still conforming to the business constraints. The main contributions of this paper are summarized as follows.

- We dissect the basic architecture of the two popular video streaming platforms by monitoring the communications between the client-side player and various components of the two platforms.
- We analyze how Netflix and Hulu make use of multiple CDNs under changing bandwidth conditions. We find that both Netflix and Hulu players stay with the same CDN even if the other CDNs may offer better video quality. In addition, Netflix tends to tie preferred CDNs with user accounts, while Hulu randomly selects the preferred CDN for individual video playback following an underlying CDN utilization distribution. Such CDN selection decisions are likely tied to—and constrained by—the business relations between the content providers and CDNs.
- We perform an extensive bandwidth measurement study of the three CDNs used by Netflix and Hulu. The results show that there are significant variations in CDN performance across time and geo-locations. These results imply that the (static) CDN selections made by Netflix and Hulu do not necessarily provide the best QoE to end-users.
- Finally, we explore alternative strategies for improving video delivery performance using multiple CDNs while conforming to the business constraints. 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 improvement over the static CDN assignment strategy currently employed by Netflix. Furthermore, using multiple CDNs simultaneously can achieve more than 50% improvement. Higher available bandwidth opens doors for supporting ever-improving video quality (thus higher video bit rate) and new services such as 3-D movies and/or multiple concurrent movies in a single household.

The paper is organized as follows. Sections II and III describe the architectures of Netflix and Hulu video streaming platforms and their CDN selection strategy. Section IV presents our measurement study of the three CDNs. Section V explores the alternative strategies for CDN assignment in order to improve video delivery performance. Section VI discusses the related work. Finally, Section VII 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 [7] 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 [8] 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* [9] to strategically throttle individual CDNs' bandwidth, we discover how Netflix makes use of multiple CDNs in the face of bandwidth fluctuation.

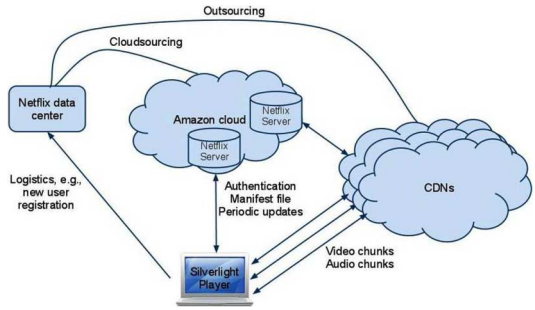


Fig. 1. Netflix architecture.

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

A. Overview of Netflix Architecture

To observe the basic service behavior, we create a new user account, login into the Netflix Web site, 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 has contacted. We also perform WHOIS [7] 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: 1) registration of new user accounts and capture of payment information (credit card or Paypal account); and 2) redirect users to `movies.netflix.com` or `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 the Netflix team [10].

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 [11]. Reference [10] indicates that Netflix uses various

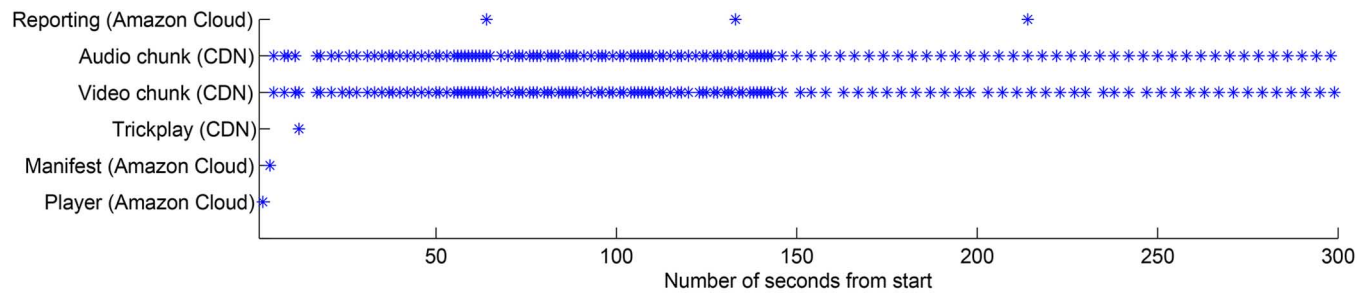


Fig. 2. Timeline in serving a Netflix client.

Amazon cloud services, ranging from EC2 and S3 to SDB and VPC [11]. Key functions, such as content ingestion, log recording/analysis, DRM, CDN routing, user sign-in, and mobile device support, are all done in the Amazon cloud.

CDNs: Netflix employs multiple CDNs to deliver the video content to end-users. The encoded and DRM protected videos are sourced in the 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 *Dynamic Streaming over HTTP* (DASH) 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 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 min, 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.

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 the "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 is 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 the 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 timeout 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.

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 s, and multiple resolutions and pixel aspects 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 4 s—the playback length of a typical chunk.

```
<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.

```
<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.

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.

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 light 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 versus unpopular, movies versus TV shows) may impact the streaming parameters. We use six different user accounts; 25 movies of varying popularity, age, and type; and 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 a 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 10 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. Such assignment of ranking seems to be independent of available bandwidth from each CDN as shown in Section IV.

2) Audio/Video Bit Rates: Netflix serves videos in multiple formats and bit rates. When a Netflix client requests for the manifest file from Netflix, the client indicates the formats of the content it can play. The Netflix server then sends back a manifest file based upon the client request. For instance, a Netflix client running on an older computer (Thinkpad T60 with Windows XP) and one on a newer computer (Macbook Pro with Snow Leopard) have different capabilities and receive different video downloading formats 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 bit rates between 100–1750 kb/s [and 2350 and 3600 kb/s for videos available in high definition (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 bit rates, whereas non-HD content can be served in up to 12 different bit rates. We also note that Netflix clients do not try all possible available bit rates when trying to determine the optimal playback rate.

D. CDN Selection Strategy

We have seen that a Netflix client can choose different video bit rates and different CDNs for video downloading. In this

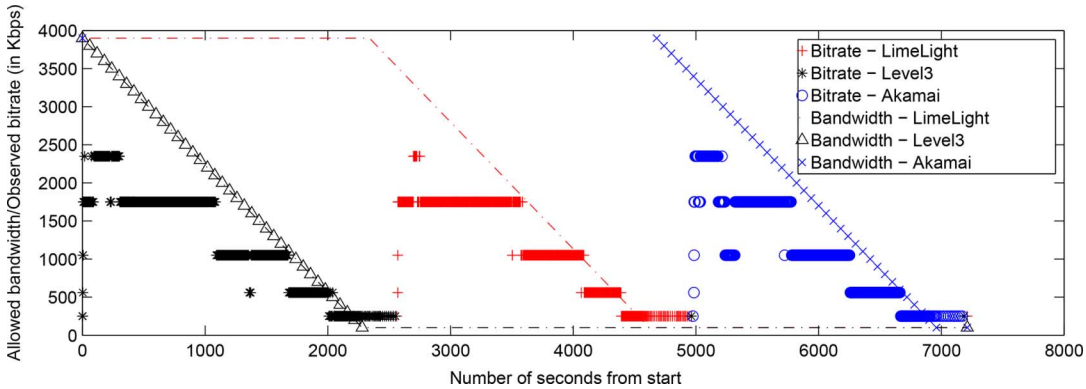


Fig. 5. CDN switching.

section, we conduct experiments to help understand how Netflix makes 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 3900 kb/s. After every minute, we reduce the available bandwidth for the current CDN by 100 kb/s until it reaches 100 kb/s. 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, and third, respectively. The client starts downloading video chunks from the first CDN. In the beginning, it starts from a low bit rate and gradually improves the bit rate 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 bit rate 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 100 kb/s), it switches to the second CDN. It repeats almost the same behavior as we leave the first CDN at 100 kb/s and gradually lower the available bandwidth for the second CDN while leaving the third CDN intact. In general, the Netflix clients stay with the same CDN as long as possible even if it has to degrade the playback quality level.

III. Hulu Video Streaming Platform

Besides Netflix, Hulu is another major OTT video service provider that does not own extensive infrastructure, yet manages to support large-scale video streaming service. Examining the similarity and discrepancy of Netflix and Hulu's respective video streaming platform and employed technologies shall shed light on the state of the art of the video streaming platform design. Unlike Netflix, Hulu offers both subscription-based service (called HuluPlus) and free service. Free service is for desktop users, while HuluPlus supports additional platforms, e.g., set-top boxes, mobile devices, etc., and offers HD video quality. Video advertisement is another major component of

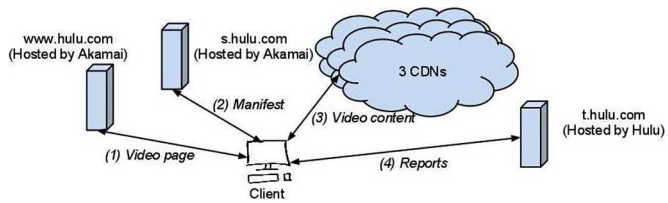


Fig. 6. High-level Hulu architecture.

Hulu, where short advertisement clips are typically delivered to users prior to the main video content.

We employ methodologies similar to the ones used in the Netflix study in studying Hulu. We play multiple videos on different Web browsers from multiple locations with different ISPs, with or without firewalls and proxy servers. We also corroborate several of these observations using what has been published on Hulu's help pages and blogs. A high-level Hulu architecture for desktop clients is shown in Fig. 6. The client gets the HTML pages for the video from Hulu's front-end Web server at `www.hulu.com`. It then contacts `s.hulu.com` to obtain a manifest file that describes the server location, available bit rates, and other details. The client uses the instruction in the manifest file to contact a video server to download the video. The client also periodically sends its status report to `t.hulu.com`. The similarity between the Netflix architecture and the Hulu architecture is striking—both platforms utilize the third-party commercial data centers and multiple CDNs to flexibly scale up/down to accommodate for the changing user population.

Bandwidth Requirements: Hulu videos are streamed at 480 and 700 kb/s. A few videos can also be streamed at 1000 kb/s. HuluPlus subscribers can also access videos in HD quality when available. Clients can switch between bit rates during playback based on available bandwidth, as we will explain later.

CDNs: Hulu employs the same three CDNs as Netflix to deliver video contents to users. Based on manifest files, a Hulu client is first assigned a preferred CDN hostname, and then uses DNS to select a server IP address.

Streaming Protocol: Hulu uses encrypted *Real Time Messaging Protocol* (RTMP) to deliver movies to desktop browsers. Hulu videos can be delivered over raw RTMP on port 1935 or RTMP tunneled over HTTP (RTMPT). Our experiments reveal

that Level3 prefers raw RTMP, whereas Akamai and Limelight prefer RTMPT. All three CDNs use RTMPT when TCP port 1935 is blocked (by a firewall for instance). HuluPlus on a desktop browser uses the same technologies and protocols. However, on mobile devices, HuluPlus uses adaptive streaming over HTTP. For instance, on `iPhone` and `iPad`, HuluPlus content is delivered over HTTP LiveStreaming technology [12]. Hulu advertisements are single .FLV files. These files are small (a few megabytes) and are downloaded over single HTTP transactions.

A. CDN Selection Strategy

Analyzing the captured packet traces, we find that Hulu uses only one CDN server throughout the duration of a video. Yet interestingly, it usually switches to a different CDN for the next video. To further understand how network conditions affect player behavior and CDN selection strategy, we conduct the same bandwidth throttling experiment as in the Netflix measurement study. At the beginning, servers from each CDN are allowed to send data at 1501 kb/s. At the end of every minute, we reduce the available bandwidth for the current active CDN by 100 kb/s until it reaches 1 kb/s. As we lower the available bandwidth for the current CDN while leaving the other CDNs intact, we notice that instead of switching to a different CDN, which is not throttled, the client keeps lowering the bit rate and stays with the original CDN. This indicates that Hulu adapts to changing bandwidth by adjusting the bit rates and continues to use the same CDN server as long as possible. Only when the current CDN server is unable to serve the lowest possible bit rate, it switches to a different CDN server. In summary, for a single video playback, Hulu's CDN selection strategy is very similar to that of Netflix.

B. User Experience Reporting

From the packet trace, we find that the Hulu player sends periodic reports to a server that includes detailed information about the status of the client machine at that time, the CDN servers for video content and advertisements, and any problems encountered in the recent past. These periodic status reports are sent to `t.hulu.com`, which maps to the same single IP address from all the locations in the US. Using WHOIS [7] queries, we learn that the corresponding IP address, 208.91.157.68, is allocated to Hulu. Examples of detailed performance information contained in the periodic reports include: video bit rate, current video playback position, total amount of memory the client is using, the current bandwidth at the client machine, number of buffer under-runs, and number of dropped frames. When the client adapts bit rate due to changing network conditions, the periodic reports also include details on why the bit rate was changed. For instance, one of the messages reads "Move up since avg dropped FPS $0 < 2$ and $\mathrm{bufferLength} > 10$," with FPS indicating *frames per second*. It appears that Hulu has sufficient user performance information for dynamic CDN selection if they choose to do so.

C. Manifest Files and CDN Usage Analysis

Hulu clients follow the manifest files they receive from the server to decide which CDN to use. Since Hulu encrypts the manifest file sent to the client, the manifest files from the network traces are not readable. We collect the manifest files using a tool called `get-flash-videos` [13]. A small section of an example Hulu manifest file is shown in Fig. 7. The last line in the figure shows Hulu's CDN preference in that manifest file. When we examine CDN preferences in a few manifest files, we observe that the preferred CDN varies from one manifest file to another. For instance, when we make two sequential requests for the same video, the preferred CDNs for those two requests can be different.

```
'title' => 'Soldier Girls',
'tp:Ad_Model' => 'longform',
'tp:Frame_Rate' => '25',
'dur' => '4991138ms',
'tp:enableAdBlockerSlate' => 'true',
'tp:Aspect_Ratio' => '4x3',
'tp:BugImageURL' => '',
'tp:researchProgram' => '',
'tp:comScoreId' => '',
'tp:hasBug' => 'false',
'tp:defaultBitrate' => '650_h264',
'tp:primarySiteChannelNielsenChannelId' => '71',
'tp:CP_Promotional_Link' => '',
'tp:CPIdentifier' => 'ContentFilm',
'tp:Primary_Category' => 'Documentary and Biography',
'tp:adType' => '',
'tp:fingerPrint' => 'csel3_prod_iad115',
'tp:CP_Promotional_Text' => '',
'tp:Segments' => 'T:00:11:54;22,T:00:26:29;09,
  T:00:38:49;27,T:00:57:37;18,T:01:03:15;02,
  T:01:17:12;03',
'tp:adTypePlus' => 'SponsoredFilm',
'tp:Tunein_Information' => '',
'tp:distributionPartnerComScoreId' => '3000007',
'tp:secondarySiteChannelNielsenId' => '38',
'tp:cdnPrefs' => 'level3,akamai,limelight',
```

Fig. 7. Section of Hulu manifest file.

To better understand how Hulu selects different CDNs, we request the manifest file every second for the same video from the same computer for 100 s. Fig. 8 depicts the preferred CDN along time, with "$*$" indicating the selected CDN for each request. Since the network conditions on the tested Hulu client is fairly stable during the experiment, the above result indicates that Hulu CDN selection is not based on instantaneous network conditions.

To further understand the impact of various factors such as client location, video, and time on CDN selection, we use the `get-flash-videos` tool to collect manifest data for 61 different videos of different genres, length, popularity, and ratings available on Hulu from 13 different locations across the US over multiple days (up to 24 days at one of the locations). The client machines on these locations are connected to residential broadband networks or business high-speed Internet services. They also cover a number of different ISPs including Comcast, AT&T, Verizon, and CenturyLink.

For a given video at a given location and time, we download the manifest file 100 times, with 1-s interval between two consecutive downloads. We call such 100 consecutive downloads an *experiment*. Each downloaded manifest file assigns one CDN as the preferred CDN. We count the number of times each CDN is preferred for each experiment. We refer to the percentage of times that a CDN is preferred in an experiment as *preference*

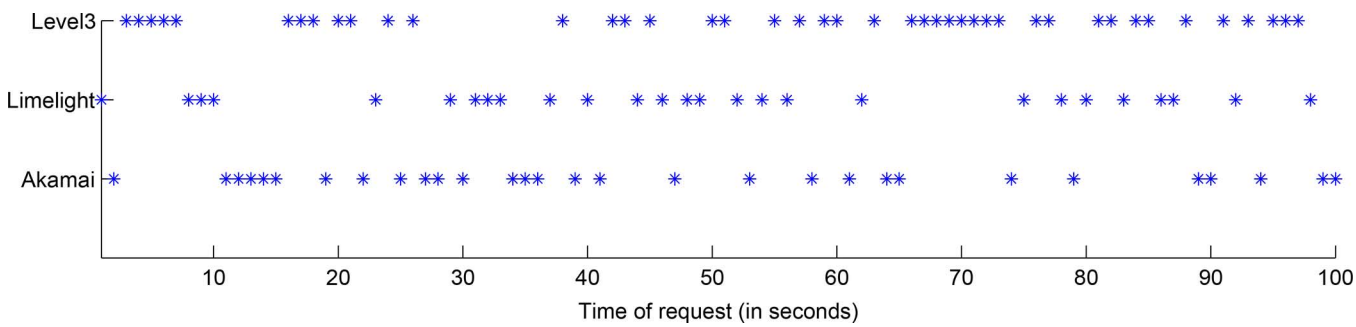


Fig. 8. CDN preference change in a 100-s time interval.

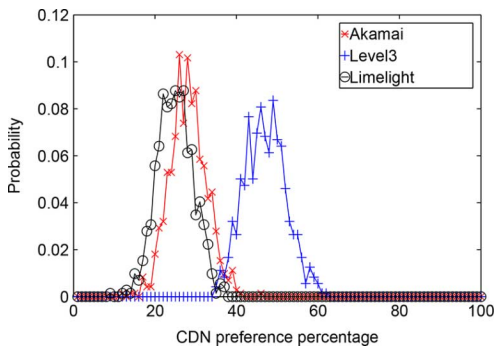


Fig. 9. Overall CDN preference distribution.

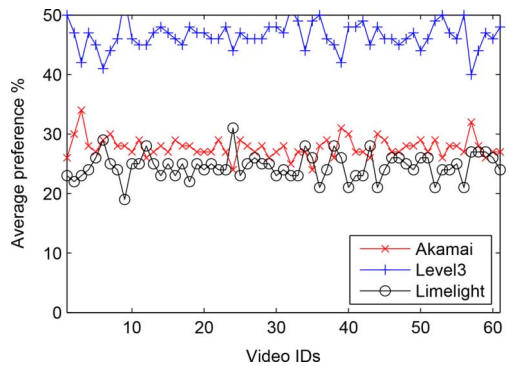


Fig. 11. CDN preference for different videos.

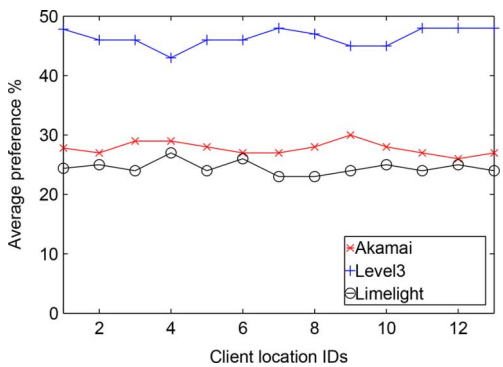


Fig. 10. CDN preference from geographic regions.

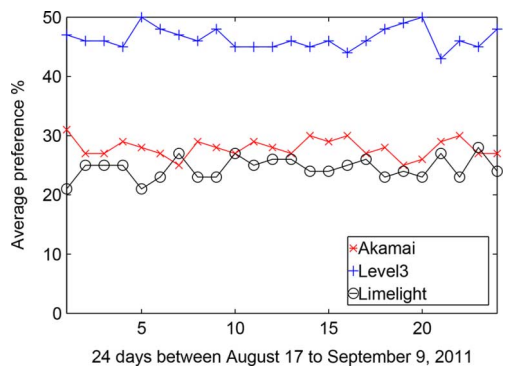


Fig. 12. CDN preference over time.

percentage. This preference percentage essentially reflects the likelihood for a CDN to be selected by Hulu.

Overall CDN Preference: Fig. 9 shows the distribution of preference percentage for the three CDNs based on results for all videos, locations, and time. The three curves representing the three CDNs are very close to Gaussian distributions. The mean preference percentage for Limelight, Akamai, and Level3 are 25, 28, and 47, respectively. Level3 is the preferred CDN 47% of times, much more than the other two CDNs.

CDN Preference Over Different Locations: Fig. 10 shows CDN preference observed from clients at different geographic locations. These 13 locations span different cities across eight US states. For this analysis, we combine data for all the videos collected at the same location and calculate the average preference percentage for each location. We observe that different CDNs have different popularity, but the popularity does not change over different locations.

CDN Preference for Different Videos: Fig. 11 shows CDN preference for different videos. Here, we aggregate the experiments for each video across location and time and calculate its average preference percentage. The small variation in preference percentage across different videos indicates CDN preference is independent of which video is being served.

CDN Preference Over Time: Fig. 12 shows CDN preference change over different days at the same location. This result is based on 24 days of experiments at a single location. Each data point represents the average preference percentage over all videos on each day for a given CDN. The results for other locations (not shown here) are similar. We observe that the CDN preferences do not change over time either.

In summary, we conclude that Hulu selects the preferred CDN randomly following a fixed latent distribution for each of the playback requests. On average, one CDN (Level3) is preferred more than others, but such selection preference does not seem to

depend on instantaneous network conditions. It is also evident that CDN selection is not affected by client location at which the video is played. Also, the selection does not change over the 24 days that we measured. We conjecture that such CDN preference is most likely based on pricing and business arrangements and is not dependent upon instantaneous bandwidth or past performance history of the CDNs. Note that the above CDN usage analysis is not doable for Netflix since Netflix ties the CDN preference to the individual user account. It is impractical to create a large number of Netflix accounts to infer its CDN usage strategy.

IV. CDN Performance Measurement

In the previous sections, we have shown that Netflix ties the CDN preference to user accounts, while Hulu chooses the preferred CDN for each video based on a latent distribution. In both cases, factors such as user geographic locations, network conditions, and requested video contents do not trigger the CDN preference change. These observations suggest that the CDN preference and selection strategies employed by Netflix and Hulu are plausibly due to *business considerations* such as business relations (e.g., pricing agreements) between the content providers and CDNs: While Netflix and Hulu employ different CDN selection strategies (one ties the preferred CDN to each user account, and the other ties to each video), both attempt to distribute and balance the video serving traffic among the CDN in accordance with certain *latent* distribution. This raises a key design *tradeoff* in CDN selection decision-making, *business constraints versus end-user QoE*, and 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 CDN selection strategy from "optimal"? Can the strategy be improved to support higher-delivery bandwidth while conforming to the business constraints?

In this section and Section V, 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 US.[1]

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 Netflix 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 is 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 bandwidths 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 s. A round is further partitioned into four "slots," with 24 s for each slot. The first three slots of each round correspond to three CDNs, respectively, and we download video chunks of size 1.8 MB. The last slot of each round is for a "joint" measurement for all CDNs, i.e., we send GET requests to the three CDNs simultaneously, each requesting video chunks for 0.6-MB 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 2 h between 8–10 pm CST, from June 8–26, 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 2-h period), and average bandwidth (over entire measurement study). These metrics allow us to examine CDN performance at multiple timescales. We 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 use five different service providers. To cover a wider range of geographic locations, we also choose 83 PlanetLab nodes spread across the US 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 Section V. We use CDNs A, B, and C to denote the three CDNs without particular order in the rest of the discussion.

A. Overall CDN Performance

Fig. 13 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. 14 shows the *cumulative distribution function* (CDF) 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,

[1]As Hulu shares the same set of CDN providers with Netflix, and since the RTMPE protocol used by Hulu is more difficult to work with than the DASH protocol used by Netflix, here we choose Netflix to conduct the CDN experiments.

Fig. 13. Best CDN at each vantage point.

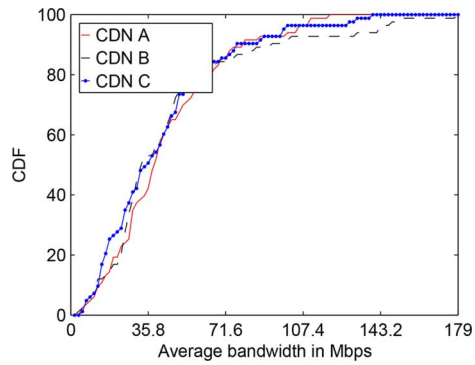


Fig. 14. CDF of average bandwidth at PlanetLab nodes.

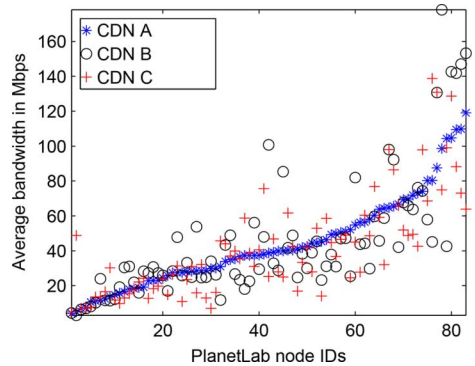


Fig. 15. Average bandwidth at PlanetLab nodes over the entire period.

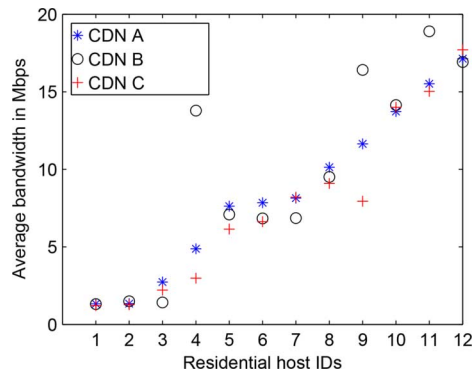


Fig. 16. Average bandwidth at residential networks over the entire period.

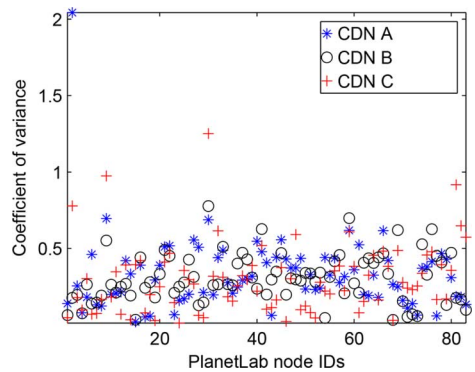


Fig. 17. Coefficient of variance for the one-day average at PlanetLab nodes.

ranging from 3 to more than 200 Mb/s. The CDF curves of three CDNs, however, are close to each other, indicating similar overall performance. Figs. 15 and 16 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., homes 1 and 2 in Fig. 16, can support 1.3 Mb/s 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 considerably. We find that these three homes use the same ISP. It is conceivable that CDN B has a better presence in this provider's network.

B. Daily Bandwidth Variations

Next, we examine the bandwidth variation at different sites from the three CDNs over various timescales. We compute the coefficient of variance (CoV) 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. 17 shows the CoV for the one-day average bandwidth at various PlanetLab nodes over multiple days. We indeed see high CoV at most nodes. The average CoV is 0.33, 0.30, and 0.30 for CDN A, B, and C, respectively. At most locations, there are significant variations in daily bandwidth for all three CDNs. We show a few representative locations in Figs. 18–20, which plot the one-day average bandwidth over the measurement period at one PlanetLab node and two residential sites, respectively. The results show significant variations of average bandwidth on a daily basis.

Figs. 18– 20 show that the performance ranking of the three CDNs also varies over time. Although the lowest CDN bandwidth across all three nodes is still above 3 Mb/s—sufficient to support SD levels—significant variations in bandwidth and varying rankings of CDNs over time suggest that further improvement in CDN selection strategies is possible.

C. Variations in Instantaneous Bandwidth

We further investigate the instantaneous bandwidth variation during 2 h 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

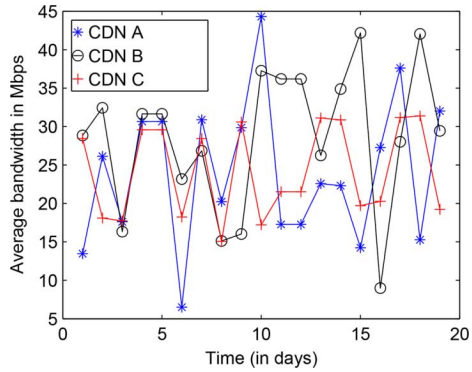


Fig. 18. One-day average bandwidth at a PlanetLab node over time.

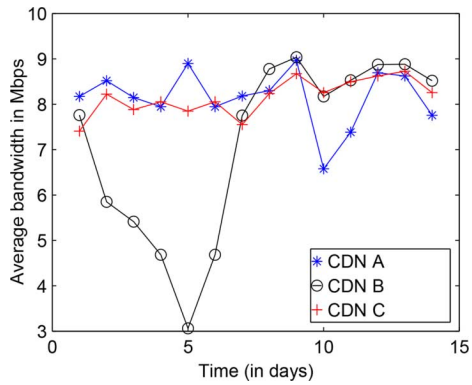


Fig. 19. One-day average bandwidth over time at residential site 7.

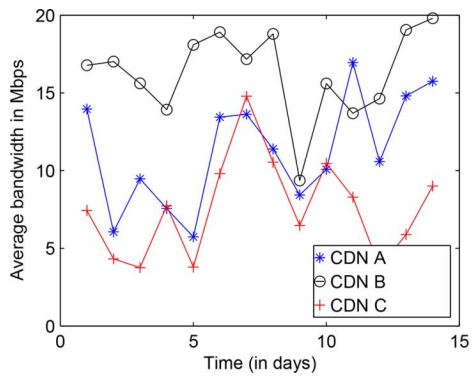


Fig. 20. One-day average bandwidth over time at residential site 9.

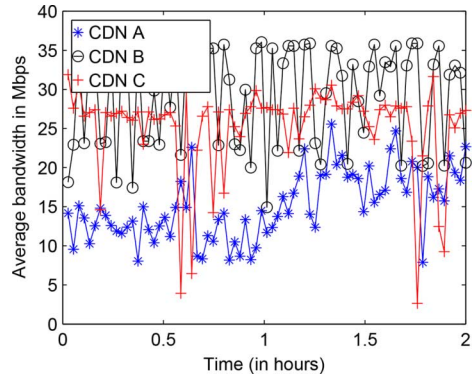


Fig. 21. Instantaneous bandwidth at a PlanetLab node.

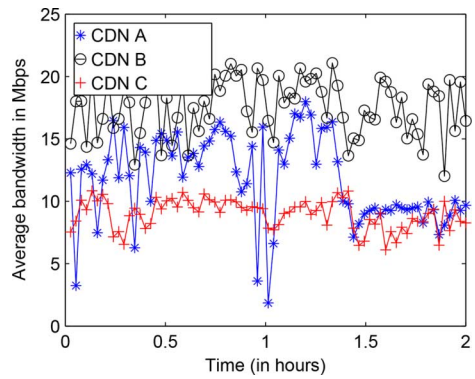


Fig. 22. Instantaneous bandwidth at residential site 7.

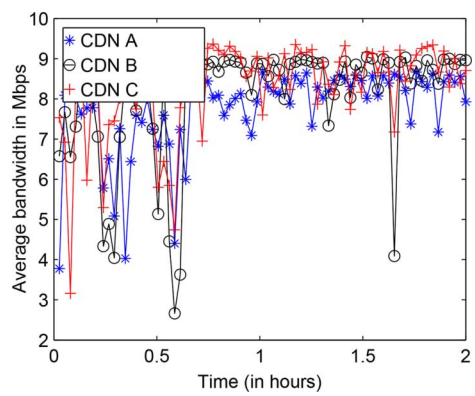


Fig. 23. Instantaneous bandwidth at residential site 9.

timescale bandwidth may significantly impact the Netflix users' viewing experience as 2 h is a typical length of a movie. Figs. 21–23 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. 21 alternates between two levels, one around 35 Mb/s and one around 20 Mb/s. The average coefficient of variation for a 2-h period is 0.19, 0.21, and 0.18, respectively, for CDNs A, B, and C for residential sites.

V. Alternate Video Delivery Strategies

We have shown that Netflix and Hulu always prefer to use one CDN for video delivery, with the other two CDNs serving as backups: They are used only if the current CDN cannot support even the lowest video quality. We have also shown that the available bandwidth on all three CDNs varies significantly over time and over geographic locations. For instance, as shown in Fig. 13, 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. Hence, if users are given a bad CDN choice, their video viewing quality may suffer even though other CDNs can provide them with a 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 the future, e.g., 3-D movies or multiple concurrent movies in the same household. In this

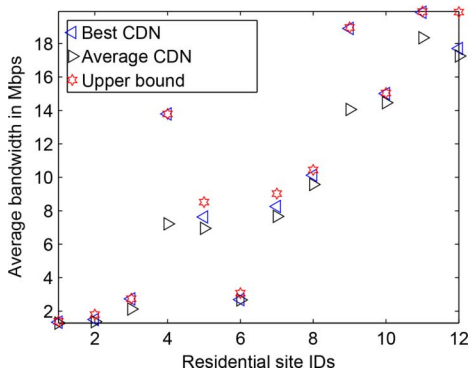


Fig. 24. Average bandwidth and the upper bound at residential sites.

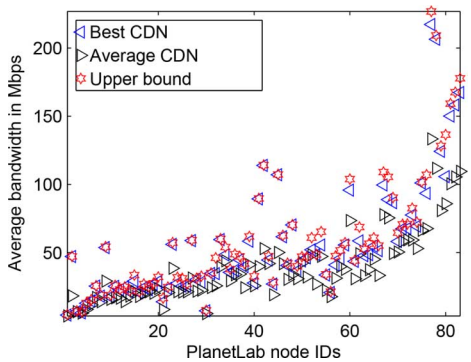


Fig. 25. Average bandwidth and the upper bound at PlanetLab nodes.

section, we first determine how much room is available 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 CDN selection strategies that can easily be deployed in practice, and demonstrate that we can indeed significantly increase the bandwidth for video delivery 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 as 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 CDN is used at any given time. We refer to the average bandwidth achieved by the optimal strategy as the *upper-bound average bandwidth*.

Figs. 24 and 25 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 the top CDN is only 6%–7% worse than the theoretical optimal case. This indicates that if we can estimate which CDN is likely to perform best in the next couple hours, we can achieve average bandwidth that is fairly close to the *upper-bound average bandwidth*.

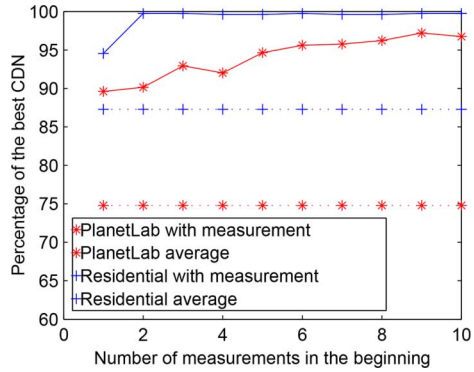


Fig. 26. Effect of number of measurements.

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. 26 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 the 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 (≥ 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 an 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

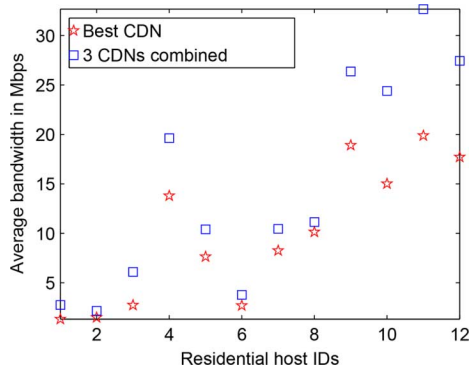


Fig. 27. Best CDN versus three combined CDNs for residential hosts.

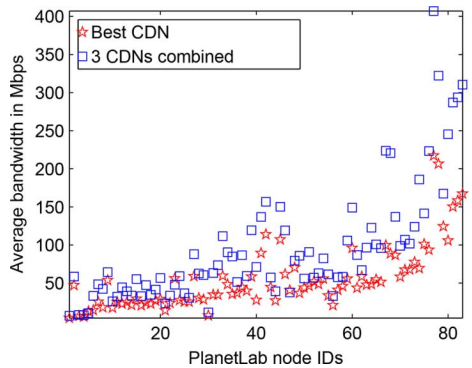


Fig. 28. Best CDN versus three combined CDNs for PlanetLab nodes.

used, whether they can offer higher aggregated throughput for end-users.

Figs. 27 and 28 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 three CDNs is greater than the bandwidth of the single best CDN by 54%–70% for residential sites and PlanetLab nodes, respectively.

D. Considering Business Constraints

In the above client-side initiated CDN selection strategies, the business constraints are not considered. We intend to show how much performance improvement can be achieved by allowing the end-users to freely choose the best CDN or use multiple CDNs simultaneously. In practice, the content providers may well have business arrangement with CDN service providers, in terms of pricing, the amount of traffic needed to be carried over a specific CDN, etc. How to integrate the business constraints with the client-side initiated CDN selection strategy to form a practical and high-performance video delivery framework is an interesting research challenge. We consider one type of the business constraints, where the overall traffic is shared by three CDNs based on a fixed split ratio—the same business constraint as used by Hulu (see Section III-C). We explore a *probabilistic CDN profile assignment strategy* that conforms with the aforementioned business constraint, yet provides end-users better QoE by allowing them to choose the best CDN from a list of candidate CDNs provided by the content provider.

Define *a CDN profile* to be a list of candidate CDNs from which the end-user can choose the best one to retrieve the video. Denote by P_i the ith profile. For three CDNs, there are seven valid CDN profiles, $P_0 = [C_0, C_1, C_2], P_1 = [C_0, C_1], P_2 = [C_0, C_2], P_3 = [C_1, C_2], P_4 = [C_0], P_5 = [C_1]$, and $P_6 = [C_2]$, where C_j denotes the jth CDN, $j = 1, 2, 3$. Note that P_0 offers end-users all three CDNs and hence is the most preferred; P_1, P_2, and P_3 offer two candidate CDNs that allow users to avoid the worst-performing CDN. For profiles P_4, P_5, and P_6, the end-users have no choice but to use the assigned CDN, which is the current strategy used by Hulu. Upon the arrival of a request, the content provider assigns the ith profile to this request with the probability p_i. The goal is to find the optimal $\{p_i^*\}$ so that the usage of profiles P_0, P_1, P_2, and P_3 is maximized while conforming with the business constraints. Denote by f_j the targeted traffic fraction for CDN j, and by $s_{i,j}$ the probability that an end-user selects CDN j as the best-performing CDN given the profile i. $\sum_j s_{i,j} = 1, \forall i$, with $s_{i,j} = 0$ if CDN j is not included in the candidate list of profile i. The optimization problem is formulated as the following linear program:

$$\max_{\{p_i\}} \left(w \cdot p_0 + \sum_{i=1}^{3} p_i \right) \tag{1}$$

subject to

$$\sum_i s_{i,j} \cdot p_i = f_j \qquad \forall j \tag{2}$$

$$\sum_i p_i = 1 \tag{3}$$

$$p_i \geq 0 \qquad \forall i \tag{4}$$

where $w \gg 1$ is the weight to favor profile 0. Constraint (2) ensures that the targeted traffic split is satisfied. Constraint (3) ensures $\{p_i\}$ is a valid probability distribution. The values of $\{s_{i,j}\}$ can be measured using the following approach. At the beginning of the service, the content provider randomly assigns N requests with profile $i, \forall i$, with N being a large number, e.g., 1000 requests. It then collects $N_{i,j}$, the number of requests that select CDN j given profile i. The estimation $s_{i,j} = N_{i,j}/N$ can be improved further over time. Also, a request's video size does not explicitly appear in the model. Since each profile will be assigned to a large number of requests/sessions, the average session size approaches its mean thanks to the Law of Large Numbers.

Next, we use a numerical example to illustrate the benefits of the proposed scheme. Suppose the target traffic split is [0.5 0.25 0.25]. The $[s_{i,j}]$ vectors for profiles 0–6 are [0.3 0.3 0.4], [0.5 0.5 0], [0.45 0 0.55], [0 0.45 0.55], [1 0 0], [0 1 0], and [0 0 1], respectively. The value of w is set to be 10. Solving the linear program using CPLEX, the optimal solution is $p_0^* = 0.625, p_1^* = 0.125, p_4^* = 0.25$, and the rest are zeros. In other words, 62.5% of the users are able to select the best CDN from three CDNs, 12.5% of the users are able to select a better CDN between CDN_0 and CDN_1, and only 25% of the users use the assigned CDN_0. The above example clearly demonstrates the possibility of integrating the business constraints into the CDN selection strategy. Note that the profiles are only used

for limiting user's choice during performance optimization; the CDNs that are not in the profile can still be used as backup. For example, users that are assigned P_4 can still switch to CDN C_1 or C_2 when C_0 is not feasible.

VI. Related Work

Several recent studies have been conducted in analyzing different aspects of Netflix video streaming. Akhshabi *et al.* [14] have studied several video streaming players including Netflix player and investigated how the streaming clients react to bandwidth changes. The measurement is done mostly from one fixed location. Recently, Huang *et al.* conducted the measurement study to examine the dynamic rate selection algorithm of Netflix, Hulu, and Vudu and proposed a technique to improve users' perceived video quality [15]. 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 [16], [17]. Krishnappa *et al.* have studied Hulu streaming with emphasis on improving performance using prefetching and caching [16], [17]. Adhikari *et al.* build a measurement infrastructure by using PlanetLab nodes with the goal to understand the YouTube system architecture [17]. Unlike our work, such works do not cover the behavior of multiple CDNs.

A large-scale video quality measurement study is conducted in [18] by examining a large number of video sessions from 91 content providers during a one-week time period. The results show that the existing video delivery fails to provide clients satisfactory quality of experience. A centralized video control plane framework is proposed to optimally select CDN for end-users and achieve significant QoE improvement. Unlike [18], our work focuses on two representative content providers, i.e., Netflix and Hulu, and dissects their architecture. Some of the measurement results presented in this paper, however, are consistent with the observations made in [18]. In addition, rather than a centralized control plane solution, a client-side initiated CDN selection mechanism is investigated in this work.

Measurement studies of CDNs such as Akamai, Limelight, and YouTube [17], [19], [20] have also been conducted, most of which focus on measurement of latency and do 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 [21], pathload [22], and FabProbe [23]. 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.

VII. Summary

In this paper, we performed active and passive measurements to uncover the overall architecture of Netflix and Hulu, and dissect several key components of these two streaming platforms. Since Netflix and Hulu use multiple CDNs to deliver videos to its subscribers, we measured the available bandwidth of employed CDNs and investigated its behavior at multiple timescales and at different geographic locations. We observed that neither Netflix nor Hulu takes into account the current network conditions when choosing a CDN. We found 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 supporting future bandwidth-demanding services.

As OTT continues to become one of the major means in delivering video content, the design of the streaming platform keeps evolving to be more scalable and flexible and provide better service. For instance, starting in June 2012, Netflix initiated its own content delivery network called "Open Connect" so that ISPs can directly connect their networks to Open Connect [24]. The Open Connect CDN allows ISPs to peer with Netflix CDN for free at common Internet exchanges or put Netflix storage appliances in or near an ISP network to save even more transit costs. The initiative has attracted some ISPs to connect to Open Connect. However, some major ISPs still refuse to connect with Open Connect due to the business concerns. For the users belonging to these unconnected ISPs, three CDNs are still used. Hence, our results/conclusions remain valid for these ISPs and their users. As the user base of OTT continues to grow, the design of the streaming platform must evolve and advance accordingly. Hence, keeping up with the new design, understanding its pros and cons, and improving upon it remain to be an interesting research topic for the future.

Acknowledgment

Part of this work was done while the V. K. Adhikari was a summer intern at Alcatel-Lucent, Bell Labs, Holmdel, NJ, USA.

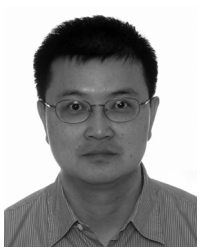
References

[1] C. Smith, "By the numbers: 20 amazing Netflix statistics and facts," May 2014 [Online]. Available: http://expandedramblings.com/index.php/netflix_statistics-facts/#.U4D7My92XsI
[2] Sandvine, "Global Internet phenomena report, Spring 2011," 2011.
[3] A. Cockroft, C. Hicks, and G. Orzell, "Lessons Netflix learned From the AWS outage," Netflix Techblog, 2011.
[4] Microsoft, "Microsoft Silverlight," [Online]. Available: http://www.microsoft.com/silverlight/
[5] V. K. Adhikari *et al.*, "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *Proc. IEEE INFOCOM*, 2012, pp. 1620–1628.
[6] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z.-L. Zhang, "A tale of three CDNs: An active measurement study of Hulu and its CDNs," in *Proc. 15th IEEE Global Internet Symp.*, Orlando, FL, USA, 2012, pp. 7–12.
[7] L. Daigle, "WHOIS protocol specification," 2004.
[8] "Tamper data," 2010 [Online]. Available: addons.mozilla.org/en-US/firefox/addon/tamper-data
[9] "Dummynet," 2013 [Online]. Available: http://info.iet.unipi.it/~luigi/dummynet/
[10] A. Cockroft, "Netflix cloud architecture," presented at the Velocity Conf. 2011.
[11] Amazon, "Amazon Web Services," [Online]. Available: http://aws.amazon.com
[12] R. Pantos and W. May, " HTTP live streaming," [Online]. Available: http://tools.ietf.org/id/draft-pantos-http-live-streaming-06.txt
[13] "get-flash-videos, A command line program to download flash videos," 2011 [Online]. Available: http://code.google.com/p/get-flash-videos/

[14] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. MMSys*, 2011, pp. 157–168.
[15] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. IMC*, 2012, pp. 225–238.
[16] D. K. Krishnappa, S. Khemmarat, L. Gao, and M. Zink, "On the feasibility of prefetching and caching for online TVservices: A measurement study on Hulu," in *Proc. 12th PAM*, 2011, pp. 72–80.
[17] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, "Vivisecting YouTube: An active measurement study," in *Proc. IEEE INFOCOM Miniconference*, 2012, pp. 2521–2525.
[18] X. Liu *et al.*, "A case for a coordinated internet video control plane," in *Proc. SIGCOMM*, 2012, pp. 359–370.
[19] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind Akamai: Inferring network conditions based on CDN redirections," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1752–1765, Dec. 2009.
[20] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid CDN-P2P: Why limelight needs its own Red Swoosh," in *Proc. NOSSDAV*, 2008, pp. 75–80.
[21] A. Downey, "Using pathchar to estimate Internet link characteristics," *Comput. Commun. Rev.*, vol. 29, no. 4, pp. 241–250, 1999.
[22] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. PAM*, Mar. 2002, pp. 14–25.
[23] 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 *Proc. PAM*, 2009, pp. 67–76.
[24] Netflix, "Netflix Open Connect content delivery network," [Online]. Available: https://signup.netflix.com/openconnect

Vijay K. Adhikari received the Ph.D. degree in computer science from the University of Minnesota, Twin Cities, Minneapolis, MN, USA, in 2012.

He is currently a Program Manager with the SQL DB team, Microsoft, Redmond, WA, USA. His primary research area is large-scale content distribution.



Yang Guo received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree from the University of Massachusetts, Amherst, MA, USA, in 2000.

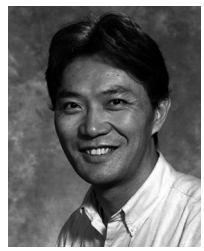
He is a Member of Technical Staff with Bell Labs Research, Crawford Hill, NJ, USA. Prior to the current position, he was a Principal Scientist with Technicolor (formerly Thomson) Corporate Research, Princeton, NJ, USA. His research interests span broadly over the distributed systems and networks, with a focus on cloud computing, software defined networks (SDNs) and its security, and Internet content distribution systems.

Fang Hao (S'98–M'00) received the B.S. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA, USA, both in computer science.

He is currently a Member of Technical Staff with Bell Labs Research, Alcatel-Lucent, Holmdel, NJ, USA. He has published about 30 papers and 20 patents for his work in networking systems, algorithms, and protocols. His research interests include software defined networks, cloud computing, routing, traffic engineering, and network monitoring.

Volker Hilt (M'13) received the M.S. degree in commercial information technologies and Ph.D. degree in computer science from the University of Mannheim, Mannheim, Germany, in 1996 and 2001, respectively.

He is leading the IP Platforms Research Group with Bell Labs/Alcatel-Lucent, Stuttgart, Germany, with a focus on cloud computing and IP communication. He joined Bell Labs/Alcatel-Lucent, Holmdel, NJ, USA, in 2002 and moved to Stuttgart, Germany, in 2012. He is a contributor to the Internet Engineering Task Force (IETF) and has published over 50 papers and coauthored more than 15 Internet drafts and RFCs. His field of research is computer networking, where he has made contributions in the areas of cloud computing technologies, distributed multimedia systems, the Session Initiation Protocol (SIP), content distribution networks, and peer-to-peer applications.



Zhi-Li Zhang (S'96–A'97–M'03–SM'11–F'12) received the B.S. degree from Nanjing University, Nanjing, China, in 1986, and the M.S. and Ph.D. degrees from the University of Massachusetts, Amherst, MA, USA, in 1992 and 1997, respectively, all in computer science.

In 1997, he joined the Computer Science and Engineering faculty, University of Minnesota, Minneapolis, MN, USA, where he is currently a Qwest Chair Professor and Distinguished McKnight University Professor. His research interests lie broadly in computer communication and networks, Internet technology, multimedia, and emerging applications.

Dr. Zhang is a member of the Association for Computing Machinery (ACM).He has co-chaired several conferences/workshops including IEEE INFOCOM 2006 and served on the TPC of numerous conferences/workshops. He is co-recipient of four Best Paper awards and has received a number of other awards.

Matteo Varvello received the Research Master's degree in network and distributed systems from EURECOM, Sophia Antipolis, France, in 2006, the M.Sc. degree (*cum laude*) in networking engineering from the Polytechnic University of Turin, Turin, Italy, in 2006; and the Ph.D. degree in computer science from Telecom Paris, Paris, France, in 2009.

He has been a Member of Technical Staff with Bell Labs, Holmdel, NJ, USA, since 2010. His current research interests include software defined networking, content-centric networking and high-speed packet classification.

Moritz Steiner received the M.S. degree (Diplom) in computer science from the University of Mannheim, Mannheim, Germany, in 2005, and the Ph.D. degree in computer networks from Telecom ParisTech, Paris, France, and the University of Mannheim in 2008. His doctoral thesis focused on P2P-based network virtual environments and large-scale P2P network measurement study of Kad.

He was a Member of Technical Staff with Bell Labs, Murray Hill, NJ, USA, from 2009 to 2013, with research interests in the areas of software defined networks, peer-to-peer networks, and cloud computing. In 2013, he joined the Web Experience Foundry team with Akamai, San Francisco, CA, USA, the cutting edge arm of the Web Experience business unit. He participates in the overall Web experience product mission by exploring new and adjacent technology opportunities.