DASHing YouTube: An Analysis of Using DASH in YouTube Video Service

Dilip Kumar Krishnappa, Divyashri Bhat and Michael Zink
University of Massachusetts Amherst
{krishnappa, dbhat, zink}@ecs.umass.edu

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) is a new streaming standard which adaptively streams video based on the link bandwidth between server and client. DASH encoded videos are chunked in small segments and each segment can have different representations. Switching between these representations enables adaptive streaming, which has the potential to reduce bandwidth consumption in cases where a video is not completely watched. In this paper, we present an analysis on the advantages and disadvantages of using DASH as YouTube's video streaming format. To perform this analysis, we make use of a YouTube video trace and analyze the potential reduction in bandwidth consumption by employing DASH in YouTube, based on user watching patterns. Results from our analysis show that by employing DASH with a segment interval of 2 seconds, we can obtain 95% reduction in bandwidth for low quality videos and up to 83% reduction for HD videos in cases where users do not watch videos completely, which is the case for $\sim 42\%$ of all video requests in our trace. Considering all videos requested in the trace the overall bandwidth reduction is 40% for low quality videos and 35% for HD videos.**

I. Introduction

YouTube is the world's most popular Internet service that hosts user-generated videos. According to [9], in 2011 YouTube had more than 1 trillion views or around 140 views for every person on Earth. Consequently, these numbers lead to a huge amount of network traffic which bears the potential of causing congestion in the network. Caching and prefetching of YouTube videos have shown to reduce bandwidth consumption and latency [17], [19], but their efficiency in reducing core network bandwidth consumption is limited. This is caused by the fact that the popularity of YouTube videos follows a long tail distribution [13], [28], and in $\sim 80\%$ of the cases a video is requested only once, which limits the effectiveness of caching and prefetching techniques. Hence, a different mechanism is required to further reduce the amount of traffic from YouTube in the network.

In today's Internet, the most popular video streaming services (e.g., Netflix, Hulu, YouTube) employ HTTP streaming over TCP, which has become a de facto standard. HTTP streaming has evolved constantly over time and can be divided into three main sub-categories. First, regular HTTP streaming, where the video is streamed or downloaded completely to the client. Second, progressive download, where the video download to the client is performed progressively but with a fixed quality. Finally, adaptive streaming over HTTP is gaining attraction since it allows to adapt the video stream's quality to the available bandwidth on the path between server and client. Dynamic Adaptive Streaming over HTTP (DASH) [27], [26] is the MPEG standard that defines this new streaming framework.

DASH, in a nutshell, divides the videos into segments with each segment representing a portion of the total length of the video. In addition, different quality versions for each segment of a video can be provided. A Media Presentation Description (MPD) file is presented to the client upon request of the video, containing information of the segments, its data ranges and the server hosting the segments. The MPD has also a detailed description of the various video bit rates or qualities offered. The client periodically (2 - 10 seconds) probes the bandwidth to the video server and switches to a different quality level based on the network condition adapting the streaming rate to the available bandwidth.

Analyzing video traffic and user behavior from a YouTube network trace collected on a U.S. university campus gateway, we find that in $\sim 42\%$ of the cases users do not completely watch the video they request. Out of the 42%, 55% of the time the viewers only watch the initial 20% of a video. By employing progressive download or regular HTTP streaming, a significant amount of data that is streamed to the client might actually not be consumed by the viewer, since this prefetched data are actually not watched if a user decides to not completely watch a video. This extra data transmitted incurs unnecessary bandwidth consumption in the network, which may lead to congestion and increase latency.

Motivated by the fact that using DASH in YouTube could significantly reduce bandwidth consumption, we investigate the advantages and disadvantages of this approach. We demonstrate the advantages of DASH by analyzing YouTube video streaming under the assumption that videos are encoded in 5 different bit rates (240p, 360p, 480p, 720p and 1080p). In the cases where videos are not completely watched (which happens 42% of the time in our trace), results from our analysis show the potential for a 95% reduction in bandwidth consumption for low quality videos (240p, 360p and 480p) and up to 83% reduction for HD videos (720p and 1080p) when using segment length of 2 seconds. We show that, considering all videos requested in the trace the overall bandwidth reduction is 40% for low quality videos and 35% for HD videos. We also show that DASH improves the Quality of Experience (QoE) of watching a YouTube video, as it switches to different bit rate streams depending on the link bandwidth to the YouTube server. Compared to the current streaming approach this will reduce the the number of pauses for buffering when watching

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

YouTube under fluctuating network conditions.

One disadvantage of DASH is the amount of storage required to host all videos in different bit rates and segment lengths on YouTube. In this paper, we take advantage of the user watching pattern and the inherent DASH feature to segment the videos to store videos distributed over several cache levels that are part of the YouTube distribution architecture. The idea is to store the first 50% of the segments of each quality version of a video in the primary caches, 50%-75% segments of the videos in secondary caches and the rest of the segments in tertiary caches. We simulate this scenario of data hosting and obtain a cache hit rate of $\sim 19\%$ by placing only 50% chunks in primary caches and thereby reducing the required cache size by half. Also, this solution reduces initial start up delays and buffering for major part of the videos.

The remainder of the paper is organized as follows: Section II presents related work in the area of DASH and YouTube bandwidth savings. In Section III, we provide an overview of the trace used in our analysis and give insight into the extra data and cost consumed by YouTube streaming techniques. Section IV demonstrates the advantages and disadvantages of using DASH in YouTube video service. Section V provides some discussion points on our analysis of using DASH in YouTube and we conclude the paper in Section VI.

II. Related Work

Adaptive HTTP streaming has been used in the Internet for quite some time. Adobe Systems HTTP Dynamic Streaming [1], Apple Inc. HTTP Live Streaming (HLS) [2] and Microsoft Smooth Streaming [6] are some of the most popular adaptive HTTP streaming solutions. Each of these adaptive streaming systems are proprietary and are not specified by an international standard.

Dynamic Adaptive Streaming over HTTP, also known as MPEG-DASH is the first adaptive HTTP-based streaming that is an international standard [26]. DASH is a relatively new standard compared to its proprietary counterparts and currently there are only a few video players with DASH implementation publicly available. E.g., DASH plugin for VLC player [25] and the DASH implementation in the GPAC project [5]. Most of the work or testing carried out on the new standard of adaptive streaming focus on different environment settings [22], [23], [24], but none of them have looked into the feasibility of using DASH for the most popular user generated video streaming service, YouTube. In this paper, we look into the advantages and disadvantages of using DASH in YouTube. The work that comes closest to ours is an investigation of peer assisted DASH by Lederer et al. [21], who look into the advantages of using P2P adaptive streaming and show 25% bandwidth reduction in CDN traffic.

YouTube is the most popular video streaming service with 100 hours of video uploaded per minute [10]. Millions of videos uploaded and billions of hours of video watched every month. Such huge amount of video requests make up large part of the world's Internet traffic, which may lead to congestion in the network. Some of the research to reduce the amount of back-end YouTube traffic involves techniques of caching and prefetching. Khemmarat et al. [17], have presented the advantages of caching and prefetching related videos offered by YouTube to reduce bandwidth and latency. Recently, we provided a new related videos reordering approach to take advantage of the videos already in the cache and show significant improvement in cache hit rate [19]. However, due to the fact that the popularity distribution of YouTube videos is long-tail, these approaches can only reduce core network bandwidth requirements up to a certain level.

Finamore et al. [15] have analyzed the network traces from campus and residential gateways and shown that users switch videos 60% of the time after watching 20% of the duration of the videos. We confirm this behavior of users switching within the 20% duration of the video from our own trace taken in a campus gateway. Although, in our trace we see only $\sim 25\%$ of the users switching within 20% of the duration of the video.

In this paper, we take advantage of this behavior of YouTube users in DASH enabled YouTube and analyze the amount of savings in bandwidth and cost. To the best of our knowledge, our work is the first to look into the feasibility of using DASH in YouTube. As we will show later in the paper (Section III), the higher percentage of users not watching more than the first 20% of a video (as reported in [15]) would actually lead to even higher bandwidth savings in a DASH enabled YouTube compared to the bandwidth consumed by the current YouTube streaming approach.

III. YouTube Data Analysis

In this section, we introduce the YouTube network trace, which we analyze to determine the user behavior in watching YouTube videos. First, we analyze in how many of the overall video requests captured in the trace a video is not completely watched. Not completely watching videos leads to the download of data that is not consumed at the client. We analyze the extra bandwidth consumption that is caused by this effect.

A. Trace Details

The network trace used in our analysis is collected with the aid of a monitoring device consisting of a PC with a Data Acquisition and Generation (DAG) card [4], which can capture Ethernet frames. The device is located at the University of Connecticut (UConn) campus network gateway, which allows it to capture all traffic to and from the campus network. It was configured to capture a fixed length header of all HTTP packets going to and coming from the YouTube domain. The trace was obtained on Feb 6th 2012 for about 72 hours and contains a total of 105,339 requests for YouTube videos. Out of these 105,339 requests, $\sim 80\%$ of the videos are requested only once during the trace period, while only $\sim 20\%$ of the videos are requested more than once. Only the latter group can take advantage of caching and consequently contribute to the reduction of bandwidth consumption and latency. Earlier research [13], [28], [17] on YouTube have shown similar trends of YouTube video popularity distribution, where few videos

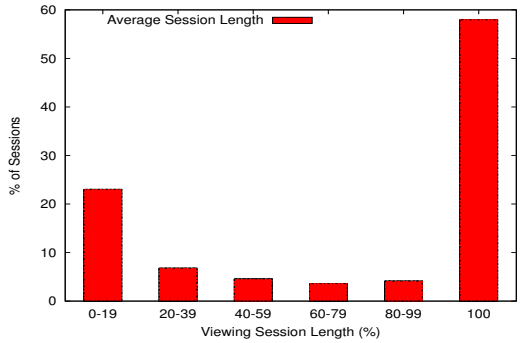


Fig. 1. Session Length for Each Request

are very popular but there is long tail of videos requested only once.

Hence, additional approaches are required to reduce bandwidth consumption in the case of YouTube videos that have very low request probability and do not benefit from caching.

B. User Watching Behavior

In this section, we analyze user behavior in watching YouTube videos by looking into the user watching pattern once the viewer selects a video. To analyze the user watching pattern, we investigate if viewers switch to a new video before they completely finish watching the current video. In order to analyze this behavior, we look into the timestamps of a user requesting two consecutive videos from the trace introduced in Section III-A. We calculate the interval between these timestamps and compare it with the duration of the initially requested video to determine if the user has switched between videos before the first video is completely viewed[1]. We perform this analysis for all the 16,284 unique users in our trace and the result is shown in Figure 1.

The figure shows the number of occurrences (in percent out of the total number of videos watched) a video is watched for $x\%$ of its total length. Results from Figure 1 show that, only in 58% of the cases videos are watched completely (this number is similar to the global study performed by Erman et al. [14]). In all other cases only part of the video is watched, with the majority of these cases ($\sim 25\%$) falling in the 0-20% session length category. This user watching pattern in switching videos without completely watching the current video results in unnecessary consumption of bandwidth[2] due to extra data downloaded to the client buffer and increases the cost of data usage for the client, which will be studied in the next section.

C. Bandwidth of YouTube Traffic

Based on the user watching pattern discussed in the previous section, we analyze the extra data traffic downloaded but not played and cost incurred by current YouTube streaming techniques in this section. We perform this analysis based on the request data obtained from the YouTube trace described in Section III-A.

[1]We ignore the last video requested by a user, as we cannot affirmatively say if it was completely watched.

[2]Hereafter, unecessary consumption of bandwidth will be referred as wastage of data or bandwidth.

Video Quality	Base Size (MB/min)	Bitrate (Mbps)	Minimum Bandwidth (Mbps)
240p	2.557	0.349	0.5
360p	4.298	0.6	0.75
480p	6.792	0.927	1.5
720p	15.487	2.114	3
1080p	32.698	4.464	6

TABLE I
VIDEO QUALITY BANDWIDTH REQUIREMENTS

The main goal is to determine how much overhead (in terms of bandwidth consumption and cost) YouTube's current streaming approach causes for the cases in which videos are not completely watched. YouTube currently offers 5 different video quality formats (240p, 360p, 480p, 720p and 1080p) and for each video request in the trace, we assume that the video is available in these 5 different formats. For our analysis, we calculate the average video size for each of the different quality formats and obtain the duration of the videos in our trace using YouTube data API [7], in addition to the information obtained from the YouTube trace.

To calculate the average video size of the different quality levels of a YouTube video, we downloaded 10 different videos, each with the 5 different quality formats offered by YouTube using YouTube downloader plugin for Google Chrome [8] (resulting in a total of 50 downloads). The second column in Table I shows the average base size[3] of a YouTube video for different quality levels for a one minute interval of the video. As expected, the highest quality video (1080p) has the highest average base size of 32.7 MB/min.

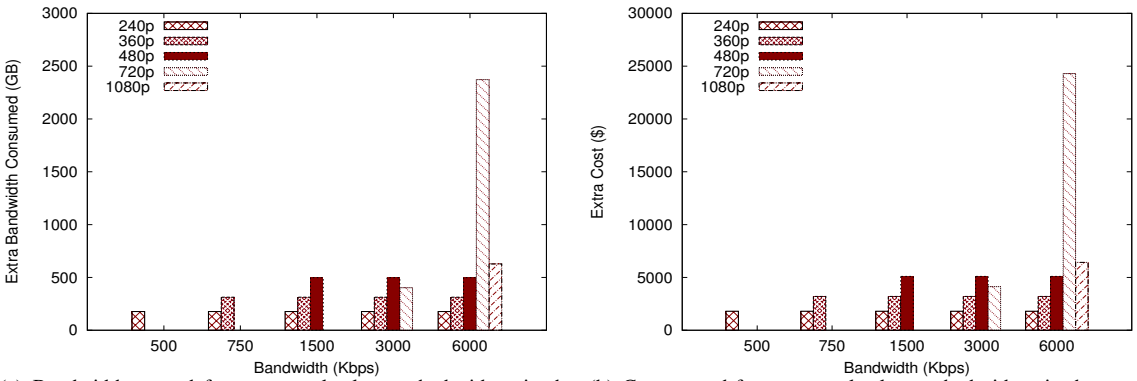
With the average base size and the duration of those 10 videos, we calculated the average bit rate of the videos for each of the qualities using

$$b = S/L \tag{1}$$

where, b is the average bit rate, S is the average base size and L is the duration of the video. The average bit rate of a video for each quality is provided in Table I. In this case, the average bit rate indicates the minimum end-to-end bandwidth required to view the video in respective quality without interruption through client buffering. The minimum bandwidth is listed in the last column of Table I and the values are in agreement with those presented in [15]. We use these values as the network bottleneck bandwidth between the client and the server to request respective video quality throughout our analysis.

Before we present our analysis results we take a more detailed look into the current YouTube streaming approaches. YouTube uses progressive download streaming for low quality videos (240p, 360p, and 480p) and regular HTTP streaming for HD videos (720p and 1080p). The difference between the two streaming techniques is that, at any given point in

[3]In this work, *base size* is the unit of data downloaded to the client.



(a) Bandwidth wasted for not completely watched videos in the trace under different network conditions

(b) Cost wasted for not completely watched videos in the trace.

Fig. 3. Bandwidth and Cost wastage for not completely watched videos due to current YouTube streaming techniques.

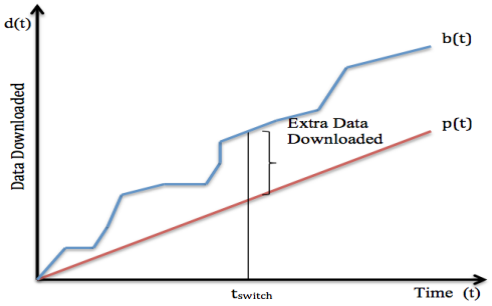


Fig. 2. Pictorial representation of extra data downloaded to client buffer at video switching instance.

time irrespective of the network connection, for low quality videos YouTube buffers only up to a maximum of 50 seconds worth of extra data from the current play point. However, for HD, YouTube employs regular HTTP streaming which downloads (buffers) the whole video to the client. In our analysis of extra data consumed by YouTube users based on user switching pattern, we apply the above mentioned approaches of YouTube streaming for the different qualities. Figure 2 illustrates the client buffer, depicting the extra data downloaded to the buffer at the time of a user discontinuing to watch a video. Line $p(t)$ shows the data consumed by the player and line $b(t)$ shows the data streamed to the client depending on the link bandwidth between client and server. The amount of data that is unnecessarily downloaded because of a viewer not completely watching a video can then be expressed as $b(t_{switch}) - p(t_{switch})$.

Figure 3(a) shows the amount of extra data downloaded but not played for each of the 5 different qualities for the minimum bandwidth cases shown in Table I. To calculate the extra data downloaded but not played, we consider only the videos which are not completely watched in our trace, as shown in Figure 1 (i.e., $\sim 42\%$ of the videos). The results show that low quality videos using progressive download streaming (240p, 360p and 480p) consume up to 600 GB of extra data as seen from Figure 3(a). It is to be noted that the extra data consumed by requesting low quality videos under higher bandwidth connection does not increase since, at any point, only 50 seconds worth of extra data is buffered. However, for HD videos, for which YouTube employs regular HTTP streaming, the extra data downloaded goes up to 2,400 GB for 720p video on a 6 Mbps network link between client and YouTube server and up to 600 GB for a 1080p video. This is a significant amount of data wasted due to extra data download and incurs additional unnecessary traffic in the network. Similarly, if we consider a network link bandwidth of greater than 6 Mbps between the client and YouTube server, the amount of data wasted increases even more than the values shown in Figure 3(a) for HD videos.

Since not all Internet services are based on the flat rate model, we also determine the extra cost the download of unwatched data can incur. This analysis is important for clients that, for example, use cellular (e.g., 3G or 4G) service to watch YouTube videos on mobiles or tablets, as the data plan cost for these services is often based on per MB downloaded. For our cost analysis, we consider a base cost of $0.01 per MB, according to AT&T's data service plan of 5 GB for $50 [3]. Our trace does not have information on the device used to request the videos, hence for our anlaysis in this section, we assume all the videos were requested via cellular devices. Figure 3(b) shows the total cost for the extra data consumed by client video requests in our trace for each video quality under different network condition. The results show that, for low quality video requests from our trace, all clients together lose about $5,000 due to extra data downloaded but not watched. For HD videos, the cost increases to $25,000. As in Figure 3(a), these values are also bound to increase for HD videos under high network bandwidth conditions.

The interesting point from Figures 3(a) and 3(b) is that these values are only for a 3 day trace from a campus with $\sim 25,000$ students. These values could increase significantly if one considers the case of regional access networks or all caches of the YouTube distribution infrastructure or if the amount of not completely watched videos is higher as reported in [15].

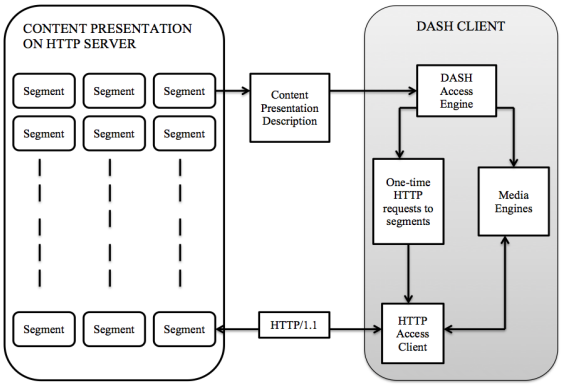


Fig. 4. DASH Flow Process

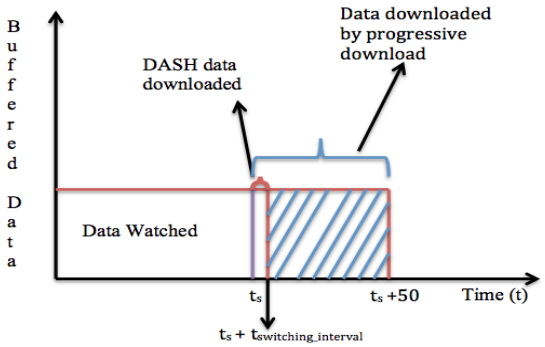


Fig. 5. Illustration of potential data savings by employing DASH.

IV. YouTube DASH

In Section III, we analyzed a YouTube trace collected from an university campus gateway. We found that YouTube users often do not watch videos completely. We also looked at the amount of data or bandwidth wasted by downloading more than the user watches and the potential related monetary loss. In this section, we will look at the new international standard in multimedia streaming, Dynamic Adaptive Streaming over HTTP (DASH), and analyze the advantages and disadvantages of deploying DASH for YouTube video service.

In the following, we provide a short overview of the functionality of DASH, then look at the amount of data and hence bandwidth savings that can be achieved by deploying DASH in YouTube. Further, we provide an insight into the hosting of DASH encoded videos in YouTube caches and show a case study on how DASH improves QoE of the viewer by reducing the number of pauses or buffering.

A. DASH

DASH [26] is the MPEG adaptive streaming standard developed for media. DASH consists of two components: A Media Presentation Description (MPD) manifest file which contains the description of the available content, their URL addresses, different bit rates or qualities available, segment lengths and other characteristics; and video segments, which contain the actual multimedia bit streams in the form of single or multiple files based on the encoder [20].

With the DASH standard, a video is encoded into different representations (e.g., different video qualities). The DASH client downloads the MPD in order to retrieve a complete list of components, including video, audio and subtitles. The DASH client then determines the preferred set of representations based on its capability (e.g., resolution), user preference (e.g., language), and network bandwidth (e.g., high or low bit rate). And finally, the DASH client downloads the desired spliced content from the HTTP server and plays the content. Figure 4 depicts the streaming flow process in DASH.

DASH videos are usually encoded with different segment lengths. Commercial approaches use segment lengths ranging from 2 seconds per fragment (e.g., Microsoft Smooth Streaming [6]) to 10 seconds per fragment (e.g., Apple HLS [2]). The segment length is an important factor when providing content for adaptive streaming. Shorter segments may provide a larger overhead in term of requests and result in more quality switches. Longer segments are not efficient in high bandwidth fluctuating environments like mobile networks, because the client cannot adjust the video stream as fast as necessary in case of a significant bandwidth change. Hence, it is important to decide on the segment lengths which suit an application or network connection best.

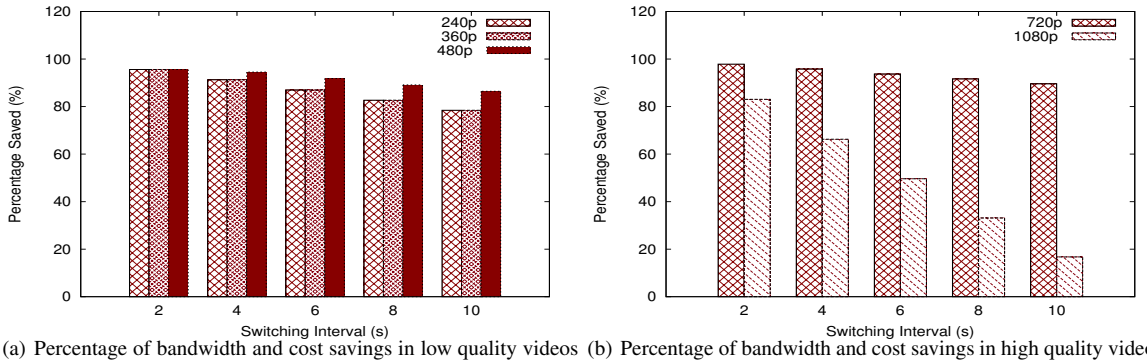
B. Bandwidth Savings

In Section III-C, we have shown that YouTube users waste bandwidth by not watching a video completely. (Obviously, this is not the case if a video is watched in its entirety.) This waste of bandwidth was also shown as potential monetary loss for mobile clients. In this section, we look into the amount of bandwidth and cost savings that can be achieved by employing DASH streaming in the YouTube video service.

Figure 5 gives an example of the amount of extra data buffered at the client by employing DASH with a switching interval[4] of $t_{switching\_interval}$ compared to YouTube's progressive download streaming approach. As shown in the figure, the amount of extra data downloaded by employing DASH at any time t_s is $d(t_s + t_{switching\_interval}) - d(t_s)$, where $t_{switching\_interval}$ is usually between 2 and 10 seconds. But with YouTube's progressive download approach, the amount of extra data downloaded is up to $d(t_s + 50) - d(t_s)$, because at any instance during video play, YouTube's progressive download strategy downloads up to 50 seconds worth of extra data to the client buffer. Hence, it is apparent that DASH saves bandwidth and cost by limiting the amount of extra data downloaded to the client. In the following, we will look at the exact amount of savings in bandwidth and cost by employing DASH.

The amount of bandwidth and cost savings that can be achieved by employing DASH in YouTube depends on the switching interval chosen. As already mentioned, a very small or a very large switching interval has its advantages and limitations. Therefore, we calculate the amount of data and

[4]Switching interval is used as synonym for segment length throughout our analysis.



(a) Percentage of bandwidth and cost savings in low quality videos by employing DASH

(b) Percentage of bandwidth and cost savings in high quality videos by employing DASH

Fig. 6. Bandwidth and Cost savings by employing DASH in YouTube for different Switching Intervals

Video Quality	Bandwidth saved for videos not completely watched	Bandwidth saved for all videos
240p	95.60%	40.15%
360p	95.60%	40.15%
480p	95.60%	40.15%
720p	97.87%	41.10%
1080p	83.06%	34.88%

TABLE II
BANDWIDTH SAVED EMPLOYING DASH WITH SEGMENT INTERVAL OF 2 SECONDS

cost savings for various segment lengths (2 to 10 seconds) for each video quality. Figures 6(a) and 6(b) show the percentage of data and cost savings for different switching intervals for low quality videos and high quality videos respectively. We performed this analysis based on the YouTube request trace presented in Section III-A.

Short switching intervals (2 seconds) yield more bandwidth savings compared to longer switching intervals (10 seconds), as longer switching intervals buffer more data compared to shorter switching intervals. Bandwidth savings for longer switching intervals depend on the switching time instance during the segment length. For example, if the user switches during the 8th second of a 10 second segment, then the amount of data transmitted but not watched is just 2 seconds, where as if the switch occurs during the 1st second of a 10 second segment, the amount of data wasted is 9 seconds. This situation is avoided in short segment length intervals as the duration between segment streams is short, the probability of data wastage during such a small interval is minute.

Results from Figure 6(a) show that, for low quality videos, DASH reduces bandwidth and cost consumption by 95% for videos that are not entirely watched under small switching intervals and about 80% for switching intervals of 10 seconds. The savings remain almost the same for each of the 3 low quality videos offered by YouTube (240p, 360p, 480p). This is attributed to the fact that YouTube already employs progressive download for these video qualities and downloads only up to 50 seconds worth of data for each of these video qualities at any point in time. In contrast, Figure 6(b) shows the savings for high quality (720p and 1080p) videos by employing DASH. The percentage of savings shown in Figure 6(b) were calculated assuming a client-server link bandwidth of 6 Mbps. By employing DASH for high quality videos, we can obtain 95% data and cost savings for 720p videos and 83% savings for 1080p videos with a switching interval of 2 seconds. These values decrease as the switching interval is increased to 10 seconds as shown in Figure 6(b). The savings shown in Figure 6(b) for HD videos increase as the link bandwidth between the client and server increases, since the extra data consumed at the client increases as shown in Figure 3(a).

Table II provides a comparison of bandwidth savings for videos which are not completely watched and for all the videos seen in our trace. Table II shows the bandwidth savings for each of the 5 different video qualities by employing DASH with a buffer interval of 2 seconds. As seen from Table II, for 240p, 360p and 480p video qualities, we obtain a 95% bandwidth savings for videos which are not completely watched, but when videos which are completely watched are considered, the bandwidth savings reduce to 40%. For HD videos, the bandwidth savings decrease from 83% to 35%, when all the videos in our trace are considered. The results from Table II show that, even while considering all the videos in our trace, the total bandwidth savings obtained are about 40% for low quality videos and 35% for HD videos when employing DASH with a segment interval of 2 seconds. This shows significant improvement in bandwidth savings when employing DASH in YouTube video service.

In summary, we have shown that employing DASH in the world's most popular video streaming service, YouTube, saves unnecessary bandwidth consumption, which has the potential to reduce congestion in the network by eliminating the transmission of unused data.

C. Data Hosting

Having shown the advantages of DASH in bandwidth and cost saving when employed for YouTube video service, we will now look into one of the disadvantages of using DASH, the additional cost of data hosting. Depending on the number of different representations provided per video, adequate storage for all representations is required.

As already mentioned, YouTube currently offers a maximum of 5 different quality versions of a video, but not all videos

# of Videos in Cache (Thousand)	Cache Size (TB)	Cache Hit Rate (%)	Partially Served (%)
20	7	30.537	53.170
40	8.59	34.690	54.294
60	8.99	36.827	54.676
80	9.2	37.766	54.772
100	9.3	38.127	54.769

TABLE III
CACHE HIT RATE FOR DASH DATA HOSTING

on YouTube are offered in these 5 qualities. According to statistics from 2011 [9], only 10% of the videos in YouTube are available in HD quality, hence only 10% of the videos on YouTube are available in the 5 different qualities offered. The remaining 90% of the videos are available in either 2 qualities (240p and 360p) or 3 qualities (240p, 360p and 480p).

Employing DASH in YouTube would increase the storage requirements on YouTube by $\sim 36\%$, since each of the billions of videos on YouTube would have to be stored in 5 different versions. This is a significant increase in data storage for YouTube as it may reduce the caching efficiency and may redirect requests to more distant (2nd and 3rd tier) caches, thus increasing the latency of serving videos to the clients and reducing the QoE of the viewer.

Our solution for this data storage conundrum lies within the YouTube data caching system and the user watching pattern. Researchers have re-engineered YouTube's data center strategy and shown that YouTube deploys a 3-tier caching strategy to serve videos to client's around the world [11], [12]. We take advantage of this 3-tier caching strategy to mitigate the DASH data hosting problem. We also take advantage of the inherent feature of DASH in segmenting the videos into small chunks and the user watching pattern in often not watching a video till the end. To avoid increasing cache miss rate and hence, increase latency in video serving from far away caches, due to increase in data hosting cost by employing DASH in YouTube, we suggest placing the earlier chunks (50%) of the videos in each of the 5 qualities in the primary caches closer to the clients and placing middle segments (50% - 75%) of the videos in secondary caches and the final segments (75% - 100%) of the videos in the tertiary caches.

We simulate the above mentioned scenario of placing the initial 50% segments of a video in the primary cache for the YouTube video requests in our trace. We store the initial segments of the video in all 5 different quality formats and calculate the cache hit rate for this scenario. The results are shown in Table III. We vary the number of videos stored in the primary cache from 20K to 100K and the resulting cache sizes required to store those partial videos are shown in Table III. Results from our simulations show that we obtain a maximum cache hit rate of 38.12% by placing only the initial 50% segments of videos in the cache. Compared to caching full-length videos, this solution reduces the required cache size by half. We also calculate the percentage of requests from the cache which are partially served, i.e., require more than the initial 50% stored in the primary cache and the results are shown in the last column of Table III. The results show that 54.76% of the requests which can be served from the cache with 100K videos (Hit Rate of 38.12%) require more than the 50% chunks stored in primary cache, which reduces our cache hit rate by 54.76%. Yet, by storing only half of the chunks of the videos and thus reducing the storage requirement by half, we obtain a hit rate of $\sim 19\%$.

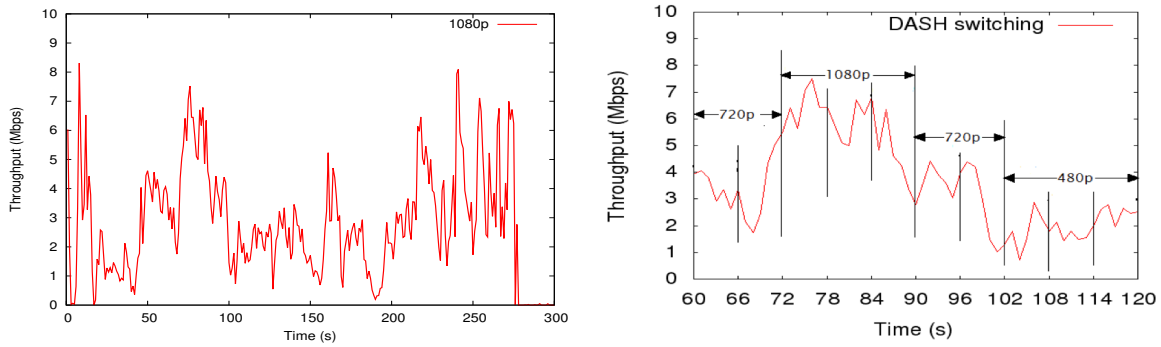
Therefore, by storing the initial 50% of the video in all available quality levels in primary caches, we can reduce the cache miss rate, reduce the storage requirement in half and reduce the latency in serving videos. In addition, this approach integrates well with the prefix prefetching approach we presented in [16]. In this earlier work we have shown that storing a 30% prefix at the first tier cache can significantly increase the viewers QoE.

D. Switching Quality

In the previous sections, we looked at the advantage of DASH in bandwidth and cost savings, the disadvantage of DASH in data hosting, and using YouTube's caching strategy and user watching pattern to overcome this disadvantage. In this section, we investigate how employing DASH improves the Quality of Experience (QoE) of the viewer. We present a case study of a viewer watching an HD video (worst case in terms of bandwidth requirements) under varying network conditions using current YouTube streaming. We determine QoE by the number of pauses encountered by the viewer while watching this HD video and show how those pauses can be eliminated by employing DASH.

We provide an illustration to show how the throughput varies at the client in the case of poor network connection while playing HD video. Figure7(a) shows the throughput per second for an HD video measured in a residential network. As it can be seen from the graph, the network condition is not suitable to smoothly play out an HD video and it takes about 270 seconds with 6 pauses to completely buffer the video of 214 seconds total length. This behavior is very annoying for a client who requests an HD video on YouTube. The client has two choices: Either to pause the video and wait for the video to buffer completely or manually select a lower bit rate quality of the video from the YouTube player settings. Recently, Krishnan and Sitaram [18] have shown that users usually get annoyed and stop watching videos even if they experience pauses or buffering for a minimum of 2 seconds. Hence, pauses during watching a video are bad for the viewer's watching experience and for the video service provider's revenue. Also, selecting a different quality manually starts buffering the video again from that point in time, removing any data already buffered, which is another disadvantage of selecting different quality version of the video.

Figure 7(b) shows the scenario in which we assumed that a DASH encoded version of the same video would be streamed. In this figure (which shows only the 60 - 120 second play out period for better illustration) we indicate the quality changes and the level of quality changes depending on the viewers link bandwidth to the server. In this example, we consider a switching interval of 6 seconds with segments of similar length



(a) Throughput variations per second while watching a 1080p video (b) Depicting DASH streaming for a 1080p video with a switching interval of 6 seconds

Fig. 7. Throughput variations and DASH switching while viewing a 1080p video.

being available in 5 different quality levels. As can be seen from Figure 7(b), the video quality switches between 480p and 1080p depending on the link throughput at the instance of the switching interval. The intervals in which the lowest (480p) quality is selected would lead to re-buffering (and consequently pause the playout) in the case of HD streaming as it is used by YouTube today.

Hence, by employing DASH in YouTube, viewers experience of watching videos can be improved due to seamless transition in quality depending on the client's link bandwidth to the video server. Also, since DASH downloads videos in short segments equal to the segment length chosen, there is no discarding of buffered data during quality switching, which saves bandwidth and cost compared to the current HTTP streaming mechanism employed by YouTube.

V. Discussion

In this section, we briefly discuss YouTube's *Auto* feature and load balancing within its cache hierarchy and the implications for our proposed approach.

A. "Auto" Feature in YouTube

YouTube offers a setting in their client player to manually choose the video quality the viewer prefers. The choices include 240p, 360p, 480p, 720p, and 1080p. Some videos might have a quality of 4096p, but this is very rare and experimental. Other than the choices of these different qualities, there is one other choice called *Auto*, which when selected changes to the quality pre-selected by the user. For example, if the user has pre-selected a quality of 720p and then selects the *auto* feature, the quality would gradually increase to 720p quality when the user selects a new video (if the video is available in that quality). This feature is close to DASH but not quite the same. DASH does not require the user to pre-select a quality to switch. The switching interval used in the *auto* feature is around 10-15 seconds and the switching is not as seamless as DASH. Another important factor is the amount of data wasted when it switches to different video quality every switching interval. The *auto* feature switches to another video quality and starts buffering data again from the point of switching during the play. Depending on the network connection, number of switches and the quality of the video at previous switching point, significant amount of data can be wasted due to this extra buffering during the play of the whole video.

B. YouTube Load Balancing of Requests

In Section IV-C, we showed the possibility of using the 3-tier YouTube caching strategy to host DASH enabled videos in 5 different qualities for each video. The MPD files for each video should therefore contain the URL of the 3 caches holding the different quality data for the video. One possible problem with that solution would be the load balancing of the requests performed by YouTube in selecting the cache to map the requests [12]. Hence, to maintain the load balancing of requests to caches and still have the MPDs for the videos sent to the client to request the videos, MPDs have to be dynamically created. This dynamic creation of the MPDs will allow YouTube to specify the host cache of a DASH video segment based on load. However, dynamic generation of MPDs should not be a problem, as web pages are dynamically generated by many applications in the Internet.

VI. Conclusion

Analyzing the YouTube user watching pattern from a university campus trace, we reveal that 42% of the time YouTube viewers rather watch only the initial part of the video than the whole. This user watching pattern leads to huge amount of unnecessary data transmission and potentially monetary loss for cellular users due to the traditional streaming techniques employed by YouTube. In this paper, we investigate the advantages and disadvantages of using DASH as a streaming standard for YouTube video service. We show that by using DASH as a streaming technique in YouTube, and encoding videos at segment length of 2 seconds, we achieve bandwidth and cost savings of 95% for low quality videos (240p, 360p, 480p) and up to 83% for HD videos (720p and 1080p) for videos not entirely watched, which is $\sim 42\%$ of the videos in our trace. When we consider all videos requested in the trace the overall bandwidth reduction is 40% for low quality videos and 35% for HD videos. One disadvantage of using DASH as YouTube streaming standard is the cost of data hosting due to the different representations of the same video. For the

data hosting problem, we suggest taking advantage of the 3-tier caching strategy employed by YouTube and the inherent feature of DASH to segment videos to store only the initial 50% of the segments of all representations in the primary cache. Hosting DASH encoded data with such a strategy provides a hit rate of $\sim 19\%$ and reduces required storage in the 1st level caches by half. Finally, we present a case study for an HD video (1080p) to show the QoE improvement of the viewer by employing DASH, due to seamless transition of video to different bit rates depending on the client's link bandwidth to the server.

VII. Future Work

As a future work on using DASH in YouTube, we would like to investigate how modifications of DASH standard for YouTube player improves the performance of YouTube and the network. Also, we would like to investigate how DASH compares to Microsoft's and Apple's dynamic, adaptive streaming approaches.

VIII. Acknowldgements

We would like to thank the anonymous reviewers and our shepherd Oliver Waldhorst for the insightful feedback on our original submission.

References

[1] Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/hds-dynamic-streaming.html.
[2] Apple HTTP Live Streaming. https://developer.apple.com/resources/http-streaming/.
[3] ATT Data Plans. https://www.att.com/shop/wireless/plans-new.html#fbid=hDVmekRQymF.
[4] Endace DAG Network Monitoring Interface. http://www.endace.com/.
[5] GPAC Project on Advanced Content. http://gpac.wp.mines-telecom.fr/.
[6] Microsoft Smooth Streaming. http://www.iis.net/downloads/microsoft/smooth-streaming.
[7] YouTube API. https://developers.google.com/youtube/.
[8] YouTube Downloader. http://www.chromeextensions.org/utilities/chrome-youtube-downloader/.
[9] YouTube Statistics. http://www.reelseo.com/youtube-statistics/.
[10] YouTube Statistics. http://www.youtube.com/yt/press/statistics.html.
[11] V. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang. Vivisecting YouTube: An Active Measurement Study. In *INFOCOM*, March 2012.
[12] V. Adhikari, S. Jain, and Z.-L. Zhang. Where Do You "Tube"? Uncovering YouTube Server Selection Strategy. In *ICCCN*, August 2011.
[13] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *IMC*, October 2007.
[14] J. Erman, A. Gerber, K. K. Ramadrishnan, S. Sen, and O. Spatscheck. Over the Top Video: The Gorilla in Cellular Networks. In *IMC*, November 2011.
[15] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. Youtube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In *IMC*, November 2011.
[16] S. Khemmarat, R. Zhou, L. Gao, and M. Zink. Watching User Generated Videos with Prefetching. In *MMSys*, February 2011.
[17] S. Khemmarat, R. Zhou, D. K. Krishnappa, L. Gao, and M. Zink. Watching User Generated Videos with Prefetching. *Sig. Proc.: Image Comm.*, 27(4):343–359, 2012.
[18] S. S. Krishnan and R. K. Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality using Quasi-Experimental Designs. In *IMC*, November 2012.
[19] D. K. Krishnappa, M. Zink, C. Griwodz, and P. Halvorsen. Cache-centric Video Recommendation: An Approach to Improve the Efficiency of YouTube Caches. In *MMSys*, February 2013.
[20] S. Lederer, C. Müller, and C. Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *MMSys*, February 2012.
[21] S. Lederer, C. Müller, and C. Timmerer. Peer-Assisted Dynamic Adaptive Streaming over HTTP System Design and Evaluation. In *Packet Video Workshop*, 2012.
[22] C. Mueller, S. Lederer, and C. Timmerer. An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. In *MoVid*, February 2012.
[23] C. Muller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer. Using Scalable Video Coding for Dynamic Adaptive Streaming over HTTP in Mobile Environments. In *EUSIPCO*, August 2012.
[24] C. Müller and C. Timmerer. A Test-bed for the Dynamic Adaptive Streaming over HTTP Featuring Session Mobility. In *MMSys*, February 2011.
[25] C. Müller and C. Timmerer. A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP. In *ACM Multimedia*, November 2011.
[26] T. Stockhammer. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *MMSys*, February 2011.
[27] C. Timmerer and C. Müller. HTTP Streaming of MPEG Media. In *Streaming Day*, September 2010.
[28] M. Zink, K. Suh, Yu, and J. Kurose. Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications. *Elsevier Computer Networks*, 2009.