YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience

A. Finamore, M. Mellia, M. Munafò
Politecnico di Torino
Email: lastname@tlc.polito.it

R. Torres, S. G. Rao
Purdue University
Email: {rtorresg,sanjay}@purdue.edu

ABSTRACT

In this paper we present a complete measurement study that compares YouTube traffic generated by mobile devices (smartphones, tablets) with traffic generated by commons PCs (desktops, notebooks, netbooks). We investigate the user behavior and correlate it with the system performance. Our measurements are performed using unique data sets which are collected from vantage points in nation wide ISPs and University campuses of two countries in Europe, and one in the US.

Our results show that the user access patterns are similar across a wide range of user locations, access technologies and user devices. Users stick with default player configurations, e.g., not changing video resolution or rarely enabling full screen playback. Furthermore it is very common that users abort video playback, with 60% of videos watched for less than 20% of their duration.

We show that the YouTube system is highly optimized for PC access and leverages aggressive buffering policies to guarantee excellent video playback. This however causes 25%-39% of transferred data to be unnecessary, since users abort the playback very early. The unnecessary data transfer is even higher when mobile devices are considered. The limited storage offered by those devices makes the video download more complicated and overall less efficient, so that clients typically download more data than the actual video length. This result is particularly critical especially for wireless networks and calls for better system optimization.

1. INTRODUCTION

Created in 2005 and bought by Google in November 2006, YouTube is the most popular and bandwidth intensive service of today Internet: it accounts for 20-35% of the Internet traffic [6, 9, 8] with 35 hours of videos uploaded every minute and more than 700 billions playback in 2010 [15, 16]. With such a high popularity, it presents a challenge both for the system itself and for the Internet Service Providers (ISP) that need to offer a good quality of service for the download streaming service. Therefore the YouTube phenomenon attracted the interest of the research community, and several works appeared to study it [7, 17, 3, 4, 1, 11, 13], focusing on either video characterization, infrastructure, or user behavior.

A second recent change in the way people access the Internet is due to the exploding popularity of mobile devices. Smartphones and Internet tablets are today commonly used both at home and at public places, and the phenomenon is still growing in popularity. Recent estimates forecast that within a few years mobile devices will be the users' preferred choice for accessing the Internet [10] while according to [9, 12] multimedia content represents a big share of the mobile traffic, with YouTube as the main contributor. Still, mobile operators are struggling with the intrinsically limited capacity of mobile access technologies.

The mix of the two phenomena has serious implications for both content providers and ISPs. Indeed, while YouTube is already commonly accessible on mobile devices from 3G/4G networks, the video encoding rate (and quality) is, by design, much more limited than the one offered to PCs. At the same time, mobile ISPs adopt tariff plans with the explicit goal to limit the amount of traffic a device can consume, a trend that is becoming popular among wired ISPs as well.

In this paper, we focus on the differences and similarities of YouTube usage when accessed from regular "PC player" or from a "mobile player". The first category includes accesses performed from a regular web browser equipped with the Adobe Flash plugin on a standard PC, e.g., a desktop, a notebook, or a netbook. The second category includes accesses performed through the special mobile version of the YouTube portal, or through the custom application found on devices running Apple iOS, Google Android, or other smartphone operating systems. By dissecting the YouTube traffic observed in operational networks, we explore the impact of devices and corresponding infrastructure synergies on the user experience and the network. Our data sets span over three different countries, including customers in both the United States and Europe, including both campus and residential networks, with very different access technologies, i.e., high speed campus LANs, WiFi hot-spots and home Access Points, ADSL and Fiber-To-The-Home links.

Our paper exposes the fact that YouTube is highly optimized to deliver video to PC users and considerably more inefficient in serving requests from mobile devices.

To the best of our knowledge, this is the first work to present an in-depth analysis of the YouTube system accessed from mobile devices. Moreover, performing a systematic comparison respect to common PCs, and taking advantage of our unique and heterogeneous data sets, we derive a comprehensive set of results. Our key findings are:

Users access content in the same manner independent of the device used:

i) content downloaded has the same characteristics both in size and duration independent of the monitored network or type of device used;
ii) the type of device does not modify the way people watch YouTube videos. Most of the watched videos are interrupted within the first 40 s, with only 10% of them lasting more than 50% of the actual video duration;
iii) people stick with default parameters, with negligible voluntary change of video resolution, and marginal fraction of views in full screen mode.

YouTube adopts different mechanisms according to the device used: the download is highly optimized for PC based players, while mobile players show consistently worse performance. This is related to the limited capabilities of the mobile devices but results shows that the mechanisms used by YouTube to serve this type of devices causes i) higher access time, ii) lower download rate, iii) more bursty traffic which impair the quality achieved by mobile players.

The amount of traffic downloaded by clients but not used by the player is large:

i) due to aggressive buffering policies, 25-39% of downloaded traffic is useless when PCs are used; this fraction grows to 35-48% when mobile devices are considered;
ii) due to a possibly unoptimized implementation in the mobile players, the amount of data transferred exceeds the actual video size for 15-25% of the download;

The last two findings are striking, and call for a possible system improvement, especially for mobile players. This is of crucial importance considering the growing popularity of mobile devices. Although our data set does not include 3G/4G mobile access clients, we believe this is even more critical for those operators for which the lack of bandwidth is and will be always a bottleneck.

The remaining of the paper is organized as follows: Sec. 2 provides a high level description of the YouTube protocols; the vantage points used are described in Sec. 3, while details on the YouTube traffic characteristics are shown in Sec. 4. Sec. 5 and Sec. 6 details the user behavior and performance, respectively. Related works are discussed in Sec. 7 before summarizing our findings in Sec. 8.

2. YOUTUBE PRIMER

In this section we provide a high level description of the protocol used to retrieve the video content. In general, two phases can be distinguished when accessing the YouTube service: 1) content look-up, 2) content download and playback. The first phase is typically performed using a web browser or a custom application running on the local client and querying regular sites (e.g., youtube.com, or a website that embeds a YouTube video). The second phase starts after the user selects the video of interest. This involves resolving the video server name for the selected video and subsequently downloading the video stream from the server. YouTube employs DNS-based mechanisms to direct clients to a server in a data center close to the user. However, in some cases HTTP-based signaling can be exploited to further redirect the user to other video servers (e.g., in the case of server congestion, or content unavailability) [13]. In this paper, we focus our analysis on the second phase.

YouTube can be accessed from a wide range of devices, each with different capabilities and hardware constraints. Depending on the client device, two mechanisms are used to retrieve the video content:

• PC-player: the client is a regular PC running either a web browser with the Adobe Flash plugin or HTML5 compliant browser [1]; we tested with several browsers and found no differences during the second phase. Hence, we will refer to them as PC-player without further distinction[2].

• Mobile-player: the client is a smartphone, a Internet tablet or a set-top-box which uses a custom application[3]. Also in this case we tested different combination of devices running both Apple iOS, Google Android and other custom operating systems. While several differences are found when considering the first phase, they all behave similarly in the second phase. Therefore, we will refer to them as Mobile-player.

Fig. 1 sketches the temporal evolution of the HTTP messages exchanged between the client and the YouTube servers. The top plot refers to the PC-player while the bottom plot refers to the Mobile-player. Clients ex-

[1]http://www.youtube.com/html5

[2]Notebooks and netbooks using regular browsers belong to the PC-player category.

[3]Even if set-top-boxes and TV appliances are hardly mobile, they use the same access mechanism than the much more widespread smartphones, and we consider them in the Mobile-player category.

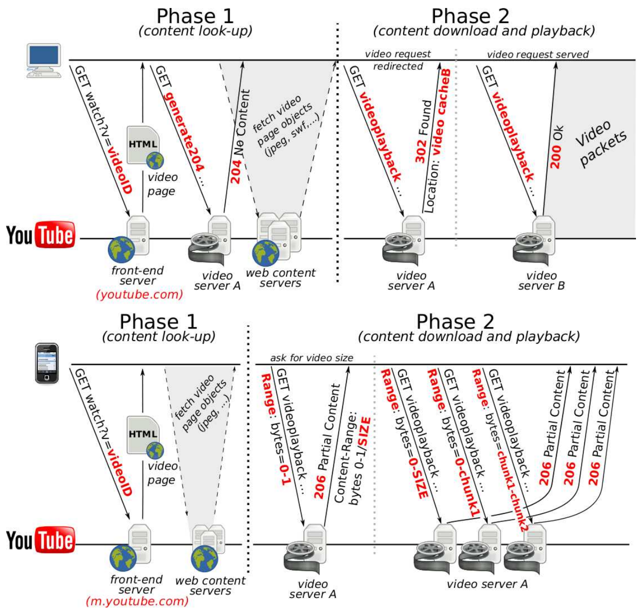


Figure 1: YouTube video download mechanisms. Example of possible evolution when accessing to *youtube.com* from a PC (top) and *m.youtube.com* from a smart-phone (bottom).

change messages either with a *front-end server*, *web content servers* or *video servers*. The front-end is responsible for the access to the YouTube portal (www.youtube.com and m.youtube.com), the web content servers provide thumbnails or other content while the video servers are in charge of the streaming. In the HTTP requests the downloaded video is identified by its *videoID* - a unique 11 characters long identifier. In the following we detail the HTTP messages that will be considered in our analysis to study the evolution of a download.

2.1 PC-player

Let us consider a client accessing the www.youtube.com web site from a regular PC using a browser as shown in Fig. 1 (top). We can split the interaction between the browser and the YouTube servers in three steps: (1) Video web page retrieval, (2) Video prefetch message and (3) Video download.

During (1), the client downloads the web page describing the video. The HTML document contains a combination of text and other "objects" (e.g., the Adobe Flash player) that the browser needs to fetch to properly display the page. Among the different objects, a javascript function triggers a `generate204` request sent to the video server that is supposed to serve the video. This starts the video prefetch (2), which has two main goals: first, it forces the client to perform the DNS resolution of the video server name. Second, it forces the client to open a TCP connection toward the video server. Both help to speed-up the video download phase. In addition, the `generate204` request has exactly the same format and options of the real video download request, so that the video server is eventually warned that a client will possibly download that video very soon. Note that the video server replies with a `'204 No Content'` response, as implied by the command, and no video content is downloaded so far.

At this point, the browser handles the control to the player which will manage the actual video download (3). The player sends a HTTP `videoplayback` request to get the video. Note that the same TCP connection previously opened during (2) can be used if HTTP persistent capability is supported between the browser and the Flash plugin. Because of server congestion or lack of content, the server can *redirect* the client to other servers [13]. In this latter case the video server replies with a HTTP `'302 Found'` response which specifies the name of another video server to contact. The player then resolves the name, and sends a new `videoplayback` request. This process can be iterated until a valid video server is found. The final video server of the chain replies with the usual HTTP `'200 OK'` response, starting the video download so that the video is locally buffered on the disk of the PC while being played on the screen.

We highlight that the `generate204` request is a specific optimization that is found only when accessing a video through the www.youtube.com. YouTube videos embedded in regular HTML pages do not exploit this, so the player have to retrieve the video server name and perform the DNS resolution before sending the first `videoplayback` request.

2.2 Mobile-player

Mobile devices use a different protocol as shown in Fig. 1 (bottom). First, no prefetch message is sent in the first phase. Second, differently from the PC-player case, the content is downloaded in "chunks", each one requested in a separate TCP connection, using the HTTP `Range` request header to specify the requested portion of the video. The video server then replies with a `'206 Partial Content'` response.

This mechanism is the result of a design choice that tries to cope with the tighter constraints in terms of storage availability for mobile devices. In fact, the mobile devices cannot buffer the entire video so the player progressively requests portions according to the evolution of the playback.

In the following sections we will investigate the impact of the different mechanisms both on the user experience and on the system infrastructure.

3. DATA COLLECTION

In this section we first introduce the tool we used to collect the traffic, giving a high level description of the mechanisms used to classify the YouTube traffic. Then,

Name	Type	Flows	Vol.[GB]	SrcIP	Videos
US-Campus	Campus	2,172,250	10,898	20,455	446,870
EU1-Campus	Campus	173,024	714	1,203	50,205
EU1-ADSL	Home	740,330	2,615	8,154	189,788
EU1-FTTH	Home	135,907	480	1,136	33,762
EU2-ADSL	Home	830,476	3,688	5,826	205,802

Table 1: Collected data sets.

we present the collected data sets that we used for the analysis.

3.1 Collection tool

To inspect the network traffic we relied on Tstat [14], an Open Source packet sniffer with Deep Packet Inspection (DPI) capabilities, which implements both traffic classifiers and fine-grained flow-level statistics. Tstat is able to rebuild TCP flows by monitoring packets that are sent and received by clients. Leveraging on this, we improved Tstat so as to identify and distinguish all possible HTTP messages that can be observed when a client downloads a YouTube video. In this paper, we focus only on the `videoplayback` and `generate204` requests from the client, and distinguish among all possible server replies (HTTP `200`, `204`, `206`, `302` responses). By parsing the URL of the HTTP messages, we can distinguish between PC-player and Mobile-player accesses[4] and extract specific video information as the videoID and the video format. Instead, from the head of the packets payload are extracted others *video metadata* as the resolution, total duration and size of the videos.

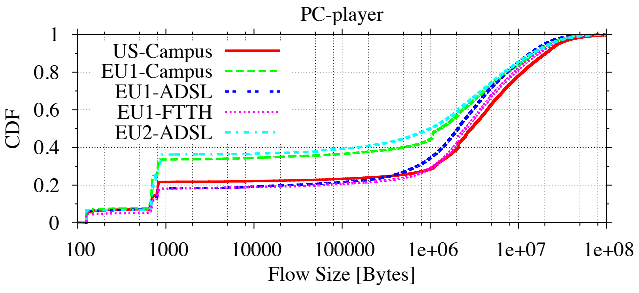
In addition to the video properties, we also collected several TCP flow-level statistics, such as the total number of packets and bytes transmitted and received, the the total flow duration and the average RTT. Further information on Tstat capabilities as well as the source code can be obtained from [14, 5].

3.2 Data sets

We collected data sets at five vantage points spread across three countries including both Points-of-Presence (PoP) in nation-wide ISPs and University campuses. These data sets represent a unique heterogeneous mix of users and technologies. At each vantage point, we installed a *probe* consisting of a high-end PC running Tstat and monitoring all traffic generated by local clients.

This paper focuses on week-long traces, collected simultaneously at the five locations, starting at 12:00 AM (local time) on February 25th, 2011. Table 1 summarizes the data sets reporting the name, the type of customers, the number of YouTube video flows and the corresponding volume of bytes, the number of distinct

[4]URL requests from mobile devices contain `app=youtube_gdata` or `app=youtube_mobile`. We do not make any further distinction on the type of browser or mobile device used since we are not interested in this information.



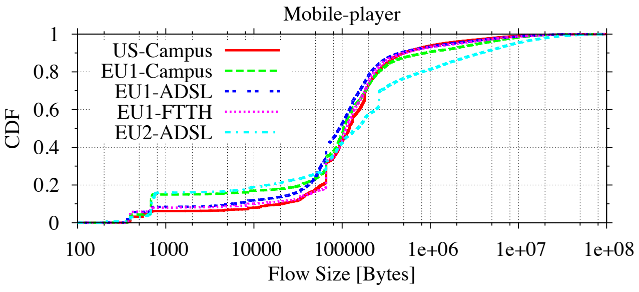


Figure 2: Distribution of the TCP flow size.

client IP addresses and the number of videos downloaded. Overall, the data sets account for more than 16 TB of traffic, more than 900,000 videos, and more than 35,000 different client IP addresses.

The Home data sets have been collected from nationwide ISPs of two different European countries. EU1-ADSL and EU1-FTTH correspond to two different PoPs within the same ISP aggregating users with ADSL and Fiber To The Home (FTTH) access technology. The campus data sets are collected in two different University campus networks, one in the United Stated and the other in Europe. To confirm the popularity of YouTube service, we observe that in all monitored networks the volume of traffic generated by YouTube videos accounts for more than 25% of the total traffic during peak time. Finally, note that the mobile traffic collected in our data sets refers to devices accessing the Internet via WiFi access networks and not via 3G ISPs.

4. FLOW AND VIDEO CHARACTERISTICS

We begin our analysis by giving an overview of the traffic generated by PC-player and Mobile-player clients. Fig. 2 reports the Cumulative Distribution Function (CDF) of TCP flow size, i.e. number of bytes (B) received by clients in a flow. Let us focus on the PC-player traffic (top plot). Steps in the CDF clearly show the presence of flows of typical size corresponding to specific HTTP messages: `'204 No Content'` flows are about 120B long, `'302 Found'` flows are in [800-1000] B range, while flows containing the `200 OK` responses are typically longer than 80 kB since they contain the video data. Interestingly, the initial part of the distribution is different for different probes, with EU1-Campus and

Name	%Flows	%Bytes
US-Campus	32.5	3.5
EU1-Campus	15.6	2.8
EU1-ADSL	27.2	3.9
EU1-FTTH	42.2	6.6
EU2-ADSL	4.2	1.6

Table 2: Fraction of flows and bytes due to mobile terminals

EU2-ADSL suffering a higher fraction of redirections (`'302 Found'`) messages. However, the tail of the distribution look rather similar, suggesting that the size of videos downloaded in the networks monitored is similar. We will detail this better in Sec. 5.

Looking at results for Mobile-player (bottom plot), we observe that the flow length is similar over different data sets, with EU1-Campus and EU2-ADSL again exhibiting higher fraction of redirection messages. However, comparing PC-player and Mobile-player we observe interesting differences: (i) the absence of the prefetching phase causes the `'204 No Content'` responses, of size 120 B, to disappear in Mobile-player; (ii) the abundant presence of the HTTP requests using the `Range` header causes the flows carrying the video data to be one order of magnitude shorter than in PC-player. This is a direct artifact of the video chunking mechanisms and not a difference in the actual video length (see Fig. 3). Interesting, the 500 B long flows are due to `'206 Partial Content'` replies to the first `videoplayback` request using the `'Range: bytes 0-1'` header that the mobile players use to discover the actual video length (see Sec. 2.2).

The effect of the chunking mechanism adopted by Mobile-player has clearly an impact on the number of flows generated by mobile devices to download the content. Table 2 quantifies this by reporting the fraction of flows and bytes that are due to Mobile-player for the different data sets. We can notice that, while Mobile-player traffic is a small fraction of the total volume, it amounts to a much larger fraction of flows. This may pose performance issues on flow-based devices, like NAT boxes or full state firewall which keep per-flow state.

Consider now the volume of bytes. Unexpectedly, only less than 6% of YouTube traffic is due to users from mobile devices. The networks we consider offer both wired and WiFi access with large penetration of smartphones, especially in the Campus networks. Therefore, one would expect that a large fraction of YouTube accesses is done from such terminals. Our measurements contrast this intuition. Moreover, some recent studies [6, 9] show that multimedia content is responsible for more than 40% of the total volume due to wireless terminals, with YouTube as the main contributor. Our results shows that this traffic is little compared to

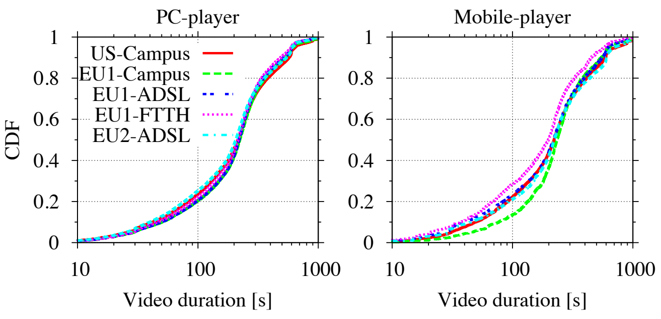


Figure 3: CDF of video duration.

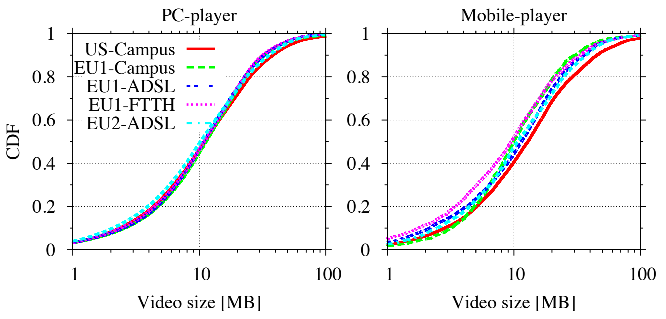


Figure 4: CDF of video size.

the volume generated by wired networks. A possible explanation for this is the fact that wireless users in our networks still prefer to access YouTube videos from standard PC browsers, including laptops and netbooks, which offer a better user experience compared to smartphones.

4.1 Video content duration and size

Fig. 3 reports the videos duration in time of the video content for both PC-player and Mobile-player on left and right plots, respectively. Note that this corresponds to the total duration not to the portion of video watched by the user. This information has been extracted from the metadata of the video streams, and is a measurement of the type of videos i) that can be found on the YouTube system and ii) that are watched by users. Consider PC-player scenario, and comparing the measurements from the different data sets. The similarity among the curves is striking so that it is impossible to distinguish between the different vantage points. For example, in all vantage points 40% of the videos last less than 3 min, with less than 5% of the videos that last more than 10 min.

Consider now the Mobile-player case. We observe a slightly large difference among the video duration accessed from different probes. Still, 40-50% of all video accessed from mobile terminals are shorter than 3 min, and 5% of video last more than 10 min. Indeed, the

ID	Video Codec	Audio Codec	Container	Res.	Name
13	H.263	AMR 8khz	3GP	144p	others
17	MPEG-4 ASP	AAC 22khz	3GP		others
5	FLV1	MP3 22khz	FLV	240p	240p-Fl
36	H.264	AAC 44.1khz	3GP		others
34	H.264	AAC 44.1khz	FLV	360p	360p-Fl*
18	H.264	AAC 44.1khz	MP4		360p-Mp+
35	H.264	AAC 44.1khz	FLV	480p	480p-Fl
43	VP8	Vorbis 96khz	WebM		others
22	H.264	AAC 44.1khz	MP4	720p	720p-Mp
45	VP8	Vorbis 128kh	WebM		others
37	H.264	AAC 44.1khz	MP4	1080p	others
38	H.264	AAC 48khz	MP4	3072p	others

(*) PC-player default format , (+) Mobile-player default format

Table 3: YouTube supported video formats.

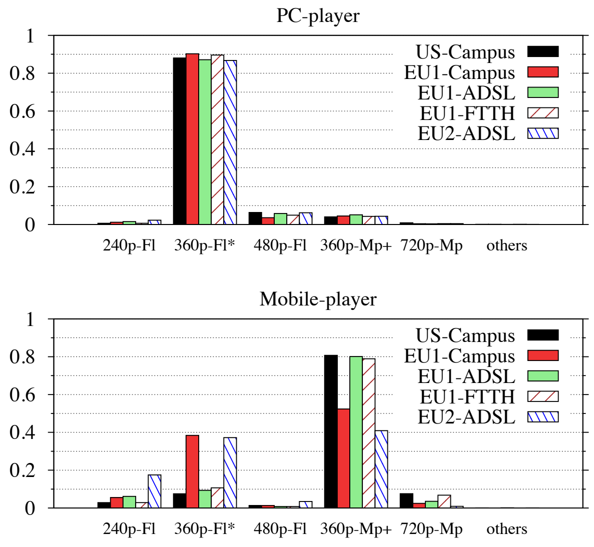


Figure 5: Fraction of videos for popular video format.

Mobile-player and the PC-player CDFs are very similar among them too. Some artifact may be also due to the smaller Mobile-player dataset (see Table 1).

This is a very strong results which shows that people with very different cultural bias (e.g., Europeans vs Americans, students/teachers vs residential users) and using very different terminals (smartphones vs PCs) and with Internet access bandwidth (ADSL vs FTTH vs WiFi vs high speed campus network) are interested in the same type of content: short videos which can be quickly watched from YouTube.

Fig. 4 reports the total video size in bytes of the videos that have been seen in our data sets. Similar consideration holds, even if one would expect the distribution to be more variable, e.g., due to the availability of videos with different resolutions, and different encoding formats. We notice that video length is very similar between Mobile-player and PC-player data sets too. The intuition would instead suggest that the video length would be larger for PC-player than for Mobile-player, if the quality of the videos watched via smartphone were assumed lower than via PC. In the following Section we dig into the impact of video codecs and resolution to give more details on these findings.

4.2 Video format characteristics and popularity

A "video" is a complex object that multiplexes encoded video and audio streams. Encoding is done according to different algorithms, and the result is then organized into a container of different type. The combination of the encoding algorithm, video resolution, and the type of container defines the *video format*. This leads to a plethora of video formats, some of which are proprietary, some others are standard.

YouTube supports the formats listed in Table 3. Each format is identified by a unique ID corresponding to the `itag` parameter in the video requests. Each ID correspond to a unique combination of video codec, audio codec, container and resolution as reported in Table 3 columns. Finally, the last column reports the name we use in this paper. A marker highlights the default video format.

The variety of formats reflects the evolution of the system and technology over the last years. In the early days, only Flash Video (FLV) content was supported with only the 240p-Fl format. In 2007 the MP4 container has been introduced along with 360p resolution. This switch was driven by the introduction of new devices that did not support the FLV container (probably Apple iOS devices). More recently, YouTube has introduced the WebM formats [15] which are part of the HTML5 specifications while the 3GP formats are specific for mobile traffic. As of today, H.264 video codec is the most adopted standard. Note that the same content is made available in different formats that are automatically generated by the system when the user upload a new video. The video format handling anyway is completely transparent to the user.

Instead, at playback time the user can eventually choose among multiple resolutions via the player user interface. For PC-player, the Adobe Flash player presents a menu button listing the available resolutions, e.g., 240p, 360p and 480p. Some Mobile-player users instead offer the choice among "good and bad" quality toggle button without the explicit indication of the available resolutions.

The supported format do not have the same popularity. Fig. 5 reports the breakdown of video format considering PC-player and Mobile-player data sets on top and bottom plots, respectively. There is a clear difference respect to the device used to access the video: Flash based formats are largely preferred by PC-player,

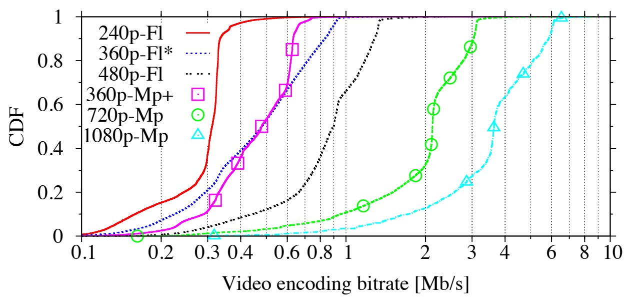


Figure 6: CDF of video encoding bitrate.

while MP4 is the preferred container for Mobile-player. This is not surprising considering that Apple iOS products (iPhone, iPad, iPod touch) cannot handle FLV content. The higher fraction for Flash based formats in the Mobile-player data sets in EU1-Campus and EU2-ADSL are possibly related to economical reasons that could motivate certain users (e.g., students) to prefer cheaper smartphones running Android O.S. or Windows Mobile which instead supports Flash content.

The default video resolution offered by the server is 360p for PC-player, while for Mobile-player the system tries to offer the best quality available according to the network/device capabilities. Indeed, the 720p (popularly known as High Definition - HD) formats are surprisingly more popular for Mobile-player than PC-player. This is the result of a design choice of the YouTube service that will be investigated in more detail in Sec. 5.

The previous findings hold true independent from the vantage point, showing the ubiquitousness of the YouTube service. We expect however this to change in 3G networks, where the 3GP formats are known to be used and low resolution videos are offered by default.

4.3 Video encoding bitrate

Given a codec and a video resolution, the video quality has a strict relation with the *video encoding bitrate*. It is therefore interesting to observe what is the typical encoding bitrate of YouTube videos. Fig. 6 reports the CDF of the video encoding bitrate for the most important video formats. Each curve aggregates statistics from all videos of the data sets (each single dataset presents the same distribution, being this a system choice). MP4 based formats are highlighted by line-points patterns. In general, the actual encoding bitrate is the minimum between the maximum allowed bitrate, and the bitrate that allows to achieve the desired quality. The latter depends on the video content, e.g., more static video sequences allow to reach lower encoding bitrate. This is reflected in the curves. For example, consider 240p-Fl (FLV) videos. The sharp knee around 300 kbps is the effect of the maximum bitrate limit, which is reached by 70% of videos. About 30% of videos is instead quality limited. Similarly, 360p-Mp (MP4) videos are configured to not exceed 600 kb/s, with most of the video that are quality limited.

As well known, the higher is the resolution, the higher is the bitrate. For example, the 360p videos (currently the default choice) do not exceed 1 Mb/s video rate, while 480p videos ranges up to 1.5 Mb/s. Up to 3 Mb/s and 6 Mb/s are required when going 720p and 1080p respectively. This allows to speculate on the impact of YouTube switching to higher resolution by default. For example a switch from 360p to 480p would correspond to almost double the amount of traffic due to YouTube, with possibly large impact on both the YouTube CDN and on ISP networks. Going to 720p would correspond to multiply by a factor of 4 the offered traffic. Given that YouTube already accounts for more than 20% of Internet traffic and assuming the users demand remains the same, this would correspond to a critical traffic surge that may impair the YouTube service too, e.g., the network cannot handle it.

5. USER BEHAVIOR AND IMPLICATIONS

In this section we focus our attention on the way people watches videos from the YouTube system, observing if they interact with the GUI, e.g., switching resolution or going in full screen mode, and which portion of the video people actually watches. Both have interesting implication on the workload the system has to handle and the efficiency it achieves in serving the requests. We first introduce the concept of *video session* which is required to characterize the user's behavior.

5.1 Methodology

As we have already seen in Sec. 4, the video download can be performed using multiple TCP connections, which is predominant for Mobile-player. This imposes to define a "download session" concept, i.e., a mechanisms to group all connections related to the download of the same content. To illustrate this, Fig. 7 shows the bitrate evolution obtained downloading the same video from a PC (top) and a mobile device (bottom) inside the EU1-Campus network. In both cases, the server starts sending an initial burst of data at a very fast rate to quickly fill the play-out buffer at the player. This is conventionally called "fast-start" mechanism. The server then starts shaping the rate as observed in [2]. Note that this is a server-based shaping mechanism in which the client has no role (neither application layer nor TCP connection control messages are sent). For PC-player, after the initial burst, the download proceeds within the same single TCP connection, whose throughput is practically equal to the average video encoding rate. Note that the average download rate is computed discarding the initial burst.

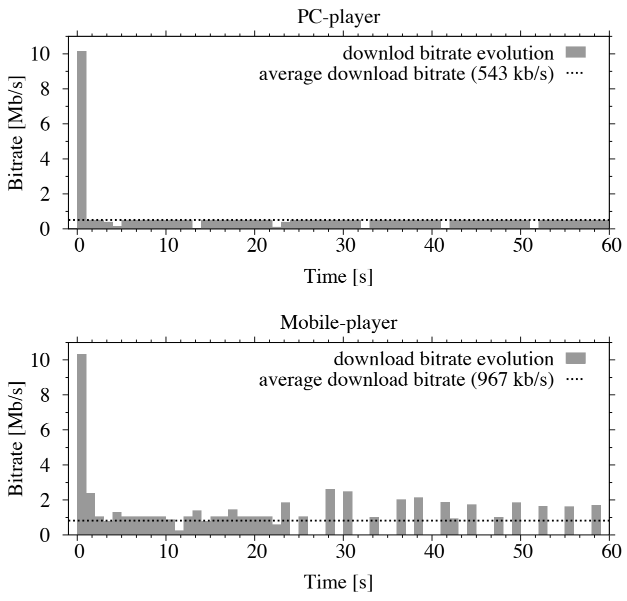


Figure 7: Example of evolution over time of the download bitrate for a video encoded at 540 kb/s.

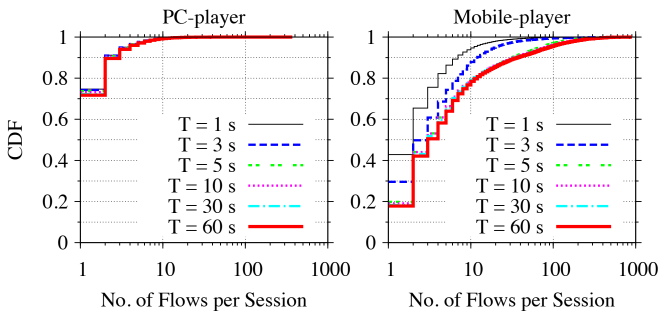


Figure 8: Sensitivity of the number of TCP connections per session for different values of T. EU1-ADSL data set.

	PC-player			Mobile-player		
Data set	0	1	>1	0	1	>1
US-Campus	95.10	4.60	0.30	99.75	0.19	0.05
EU1-Campus	96.62	3.12	0.27	99.28	0.61	0.10
EU1-ADSL	95.27	4.45	0.28	99.63	0.28	0.09
EU1-FTTH	95.73	3.99	0.28	99.39	0.42	0.19
EU2-ADSL	95.14	4.40	0.46	98.07	1.36	0.57

Table 4: Percentage of resolution switch.

For Mobile-player instead, the bitrate evolution is more bursty. This is a consequence of leveraging different TCP connections to download chunks of video. Indeed from second 23 and on, the mobile terminal aborts the ongoing TCP connection, and starts requesting chunks of video on separate TCP connections. They last about 1 second and are separated in time by about 2 seconds of silence. Since a new TCP connection is used, the server enters the "fast-start", which is early interrupted by the client which aborts again the underlying TCP connection. We believe this mechanisms is due to a client-side buffer management policy which abruptly interrupts the TCP connection when the play-out buffer is filled up. The client then re-starts the download when the buffer depletes below a certain threshold. This results in an inflation of TCP connections, and a possible inefficient download.

The early abortion of the TCP connection can be due to other causes as well. For example, a resolution change or a fast forward in the video are handled by aborting the previous download and starting a new one for both PC-player and Mobile-player. Finally, the initial control messages possibly sent on separate TCP connections are also fundamentals to capture the dynamic of the download.

We thus aggregate TCP connections in *video sessions*. Each video session corresponds to the set of connections that i) share the same source IP address and ii) same *videoID*, iii) and are separated by a silence shorter than T seconds. I.e., two connections c_1 and c_2 belongs to the same session if the time before the start (time of TCP SYN packet) of c_2 and end (time of last packet) of c_1 is smaller than T.

Fig. 8 reports the number of connections per session for different values of T. The EU1-ADSL data set is considered, the others showing identical results. The choice of T is not critical for PC-player, while $T > 5$ s is required to properly aggregate Mobile-player connections. In the following, we set $T = 60$ s, a conservative choice to better capture user's actions that could happen after the download has been completed but while the playback is still running.

Fig. 8 shows also the impact of the Mobile-player mechanisms. In fact, for PC-player, only 2% of the sessions have more than 6 connections, while for Mobile-player more than 4% of the sessions involve more than 100 connections.

5.2 Resolution switch

Given the above definition of a session, a change of video resolution is easily detected by observing requests with the same *videoID*, but different video format.

Table 4 reports the percentage of sessions involving zero, one or more than one resolution switch for both PC-player and Mobile-player. Surprisingly, results show that this happens only for less than 5% of PC-player sessions: users stick with the default video format. This means that the users are not interested in this feature (or they are unaware of it). For Mobile-player the choice of resolution is either hidden or not available, and a marginal fraction of users exploit it.

Data set	Low-to-High	High-to-Low
US-Campus	95.7	4.3
EU1-Campus	86.1	13.9
EU1-ADSL	93.9	6.1
EU1-FTTH	90.5	9.5
EU2-ADSL	83.6	16.4

Table 5: Percentage of resolution switch breakdown for PC-player.

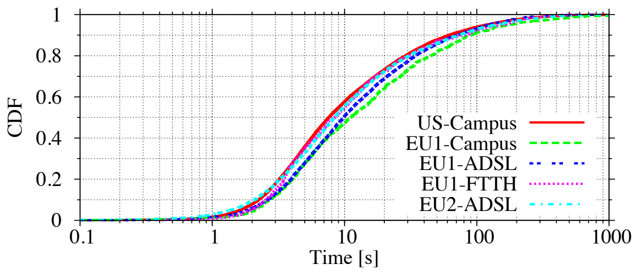


Figure 9: Time at which Low-to-High resolution switch happens.

We can further classify the resolution changes in "Low-to-High" and "High-to-Low" considering the involved video resolution. Table 5 reports the breakdown. We can see that Low-to-High are largely predominant, with the majority of them being 360p-Fl→480p-Fl switch. Interestingly, this resolution switch is triggered automatically by the player when the full screen playback is enabled (the converse is not true). Combining Tables 4 and 5 we can conclude that full screen mode is not popular, but it is the main cause of resolution switch.

As final note, the largest majority of High-to-Low switch are 360p-Fl→240p-Fl. This suggests that those are triggered by the user because of bad performance too. EU2-ADSL and EU1-Campus shows a slightly larger High-to-Low switch fractions. As we will see in the following, those are the two vantage points with slightly worse performance.

To complete the analysis, we investigate when the resolution switch is triggered. Fig. 9 shows the CDF of the time between the session start and Low-to-High resolution switch. Due to buffering at the player, this is an overestimate of the actual switch time. 50% of these events happens in the first 10 seconds, while only 10% of users trigger them after 1 minute. In terms of video size, more than 80% of the switches happens in the first 20% of the video length while only 5% occurs in the second half of the video. The same consideration holds for High-to-Low changes. Overall we can conclude that resolution changes are usually performed at the very beginning of the playback.

Amazingly, results are practically identical in all data sets despite they include very heterogeneous users habits and cultures.

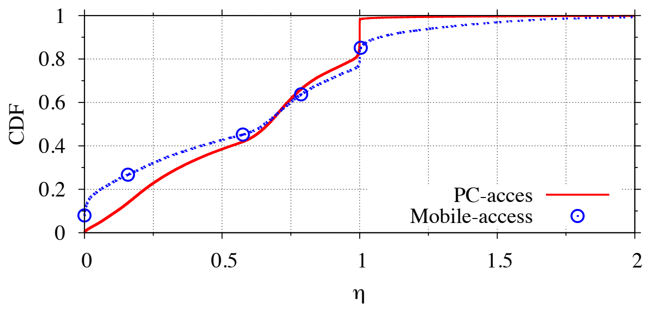


Figure 10: CDF of the fraction of downloaded video bytes. EU1-ADSL data set.

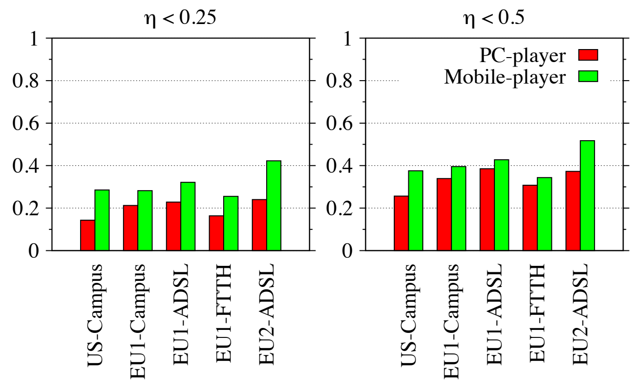


Figure 11: Fraction of video downloads having $\eta < 0.25$ and $\eta < 0.50$

5.3 Fraction of watched video

We now focus our attention on the time a user spends watching a video. To measure this, we exploit the fact that the player aborts the video downloads if the user changes web page on the browser (or custom player). Let η be the fraction of downloaded video with respect to the video length. If $\eta < 1$, then the user did not watch the entire video[5].

Fig. 10 shows the CDF for η for EU1-ADSL data set. Two considerations hold: first, about 80% of video sessions are abruptly interrupted. Second, Mobile-player results show that the player can download *more* data than the video length. We will investigate this better in the following.

To better compare results, Fig. 11 details the fraction of video downloads having $\eta < 0.25$ and $\eta < 0.5$. Interestingly this metric is very similar for all vantage points, with users on Mobile-player consistently aborting earlier then users on PC-player. Fig. 12 shows the absolute and relative time at which the user stops watching the

[5]By checking the `Range` of requests, we filter out those sessions in which the user fast-forward the playback to a position outside the already buffered portion of the video.

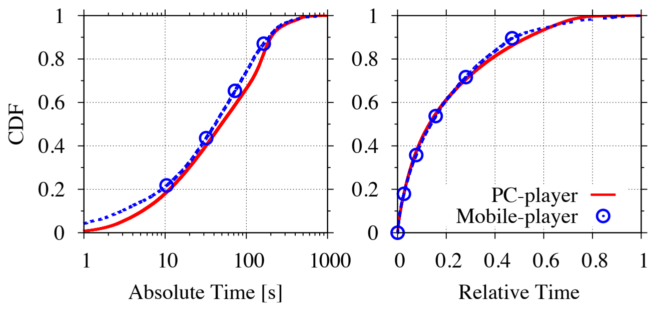


Figure 12: CDF of absolute (left) and relative (right) portion of watched video for session with uncompleted download. EU1-ADSL data set.

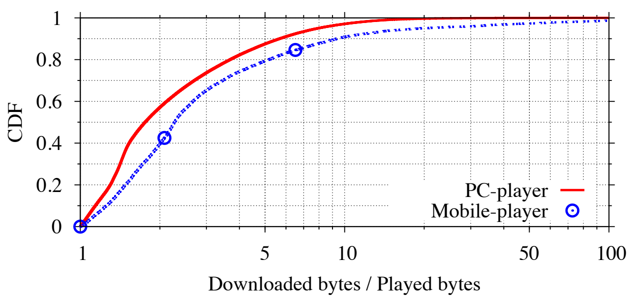


Figure 13: CDF of ratio between downloaded bytes and played bytes. EU1-ADSL data set.

video on left and right plot, respectively. It shows that people tends to abort the playback very soon, with 60% of videos that are watched for less than 20% of their duration. This can be due to a mismatch between the users' interests and the content they find on YouTube. Notice that this is also an interesting fact that could be exploited to better handle the content distribution among the CDN nodes, e.g., caching only some portion of the videos. The impact of Mobile-player versus PC-player is very limited, testifying that the probability of aborting the playback is not biased by the device.

5.4 Impact of buffering policy and user early abort

Consider now all video data already buffered at the player at the time the user aborts the playback. That data has been downloaded in vain. Fig. 13 precisely quantify it by reporting the amount of downloaded bytes over the among of data possibly consumed by the player. The latter is evaluated assuming that the playout started immediately after the first byte has been received, and that data is consumed at the video encoding bitrate. Since the initial buffering is neglected, the actual waste of data is higher than this. Results are dramatic for PC-player: 40% of sessions download more than two times the amount of data that was watched. This is

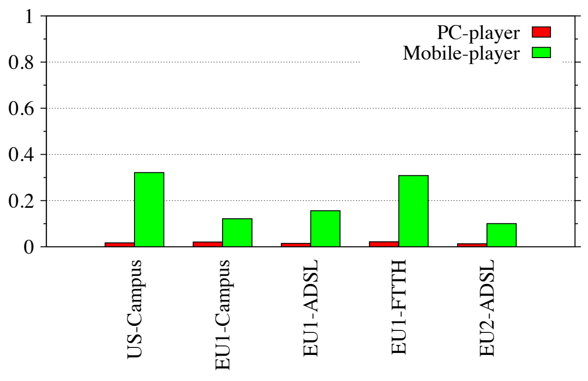


Figure 14: Fraction of sessions downloading more than the entire video.

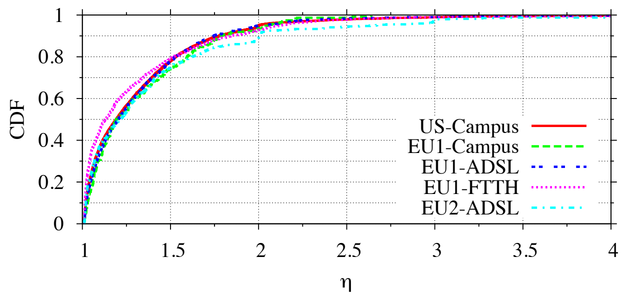


Figure 15: Ratio of downloaded data versus video length for session with $\eta > 1$. Mobile-player.

the result of aggressive buffering policies adopted by YouTube servers (recall that server-side shaping is adopted). Even worse, the Mobile-player waste is higher, with 20% of sessions downloading more the 5 times the amount of watched data.

This waste could be reduced by limiting the buffer size at the player, e.g., by implementing a more careful control based on a player-side management policy.

5.5 Sessions downloading more than video length

Let us now focus on the session for which $\eta > 1$. Intuitively, those should be very limited, since one would expect that the player should not download more data than the total video length. Fig. 14 shows the fraction of session for which this happens. Only sessions with no switch of resolution are considered. For PC-player, less than 2% of session show this. We have found that the exceeding amount of volume is possibly related to users watching the same video multiple times causing the player to re-download the video. Overall, this effect is marginal.

For Mobile-player instead we observe that 15-30% of sessions downloads more than the video size. Performing some active experiments, we have confirmed at

Data set	PC-player	Mobile-player
US-Campus	39.17	47.9
EU1-Campus	36.91	38.1
EU1-ADSL	24.93	38.7
EU1-FTTH	38.43	53.5
EU2-ADSL	29.27	35.6

Table 6: Average percentage of wasted bytes considering peak hour with respect to useful data.

least two causes for this: 1) in case of backward seeks, the player can re-download the same content because it has been already discarded from the player local buffer. This does not happen for PC-player. 2) the aggressive chunk-based download mechanisms is source of inefficiency: the Mobile-player often requests chunks bigger than needed, i.e., requesting from desired position x up to the end of the video. The server then starts sending data from x at a high rate, quickly filling up the application-layer player buffer with data up to y. This in turn causes the abortion of the underlying TCP connection, stopping the download. However TCP had already received some data at the transport-layer receiver buffer up to $y' > y$, which is discarded. The player then start asking data from y and not from y'. The aggressive server buffering policy coupled with player limited buffering capabilities is thus origin of inefficiency.

To quantify the waste of traffic due to this, Fig. 15 reports the CDF of the ratio of downloaded data versus video length for sessions with $\eta > 1$. 50% of sessions downloads 25% more data, and 4% of the sessions downloads more than the twice of the video size!

5.6 Video wasted data

Table 6 quantifies the overall percentage of wasted bytes with respect to useful data. It includes both the effect of aggressive buffer management and of chunk based video retrieval mechanisms. Measurements refer to the peak-hour time, when YouTube traffic peaks to several hundreds of Mb/s in most vantage points. Results show that the amount of traffic downloaded by clients but not used by players is comparable with the useful data traffic. For example, for US-Campus, the wasted traffic in a single hour amounts to 28.8GB and 1.5GB for PC-player and Mobile-player, respectively, corresponding to more than 7Mb/s of traffic. This corresponds to a quite large amount of wasted bandwidth both from the point of the operator and of the YouTube CDN.

We have performed experiments on Mobile-player connected to a 3G network. The problem shows up exactly in the same way, with clients downloading lot more data than the video played and length. This is a issue that is particularly critical given the increasing popularity YouTube access from mobile devices in 3G networks.

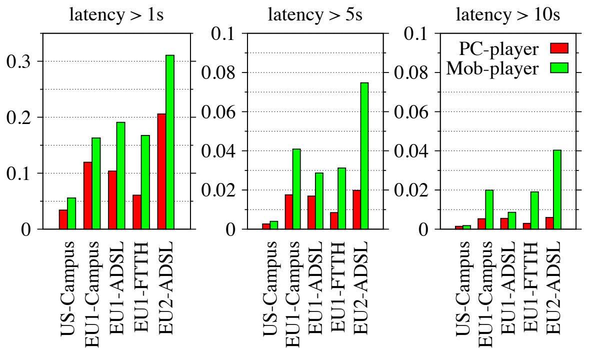


Figure 16: Fraction of sessions with high startup latency.

6. STREAMING PERFORMANCE

In this section we focus on the streaming performance considering two metrics that reflect the user experience: *startup latency* and *bitrate ratio* between download rate and video encoding bitrate. We then pinpoint possible causes that can impair them.

6.1 Startup latency

We define the startup latency as the time elapsed between the first `videoplayback` request and the 1st video packet received. This corresponds to a lower bound of the delay experienced before the actual video playback starts since the initial buffering time is ignored. The latter indeed is hard to know and player dependent. We prefer thus to focus on a simpler, more precise and accurate measure.

Fig. 16 reports the fraction of sessions with the startup latency higher than a certain threshold. Given that we are interested in studying the user experience, we selected threshold values that can be appreciated by the user, i.e., 1, 5 and 10 s. Results show that the performance is heterogeneous across the data sets, with Mobile-player suffering larger delays. We found that the delay is due to a combination of causes.

Redirections: Video requests can suffer from a different number of redirections. Each redirection involves i) a DNS query to resolve the hostname of the next video server, ii) the opening of a new TCP connection, iii) a new video query. The network distance between the client and the server plays also a significant role, since YouTube CDN is likely to direct clients to video servers with the closest RTT. In case of redirection, the server will therefore be the not preferred one.

Fig 17 reports the fraction of sessions affected by redirections. More than 70% of PC-player session does not suffer from redirections in all data sets, while Mobile-player sessions are more likely to be redirected. Understanding why this is happening is difficult. A possible cause of redirection is due to cache miss. [13] shows that after

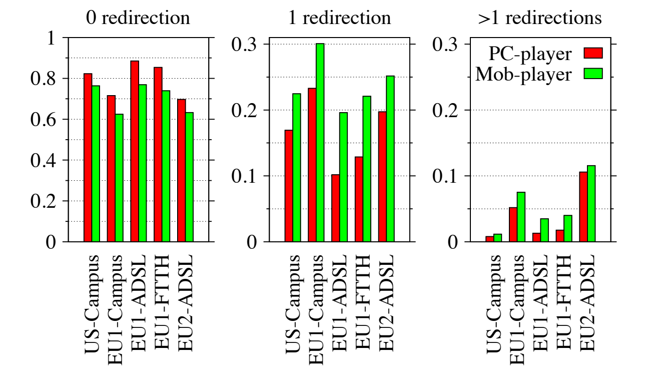


Figure 17: Fraction of sessions suffering redirections.

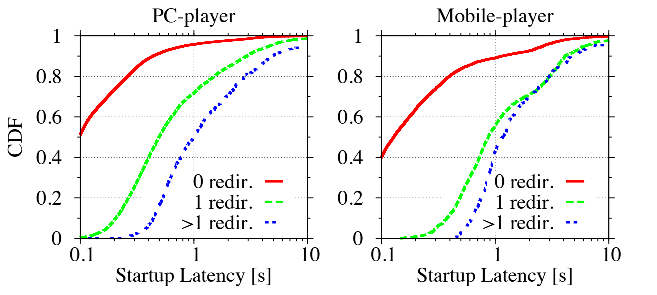


Figure 18: CDF of startup latency for different number of redirections. EU1-ADSL data set.

a cache miss at the closest data center, the following requests for the same content are directly served by the first server. This hints to a caching mechanism based on pull schemes. Since Mobile-player videos are less popular (also because of the different video format adopted by Mobile-player) a cache miss is possibly more likely to happen. Thus more redirections are suffered.

Fig. 18 depicts the impact of the redirections on the startup latency. We can see that the higher is the number of redirection in the video session, the higher is the startup delay. 20% of sessions with more than 1 redirections have a startup latency higher than 3 s for both PC-player and Mobile-player.

Video request processing time: Another possible cause of large startup time can be due to the server processing time, i.e., time needed by the video server to elaborate a video request. To estimate it, we compute the time between the last `videoplayback` requests sent by the client and the first video packet sent by the server. To eliminate the network delay we subtract the RTT.

Fig. 19 reports the CDF of the estimated processing time for both PC-player and Mobile-player in the EU1-ADSL data set. Other data sets show similar trends. We can see that 50% of the requests are served within

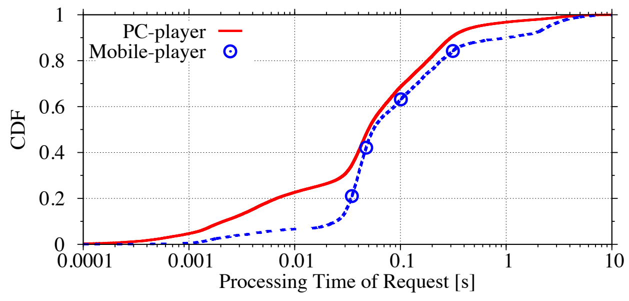


Figure 19: CDF of the video query processing time. EU1-ADSL data set.

< 50 ms. A sharp knee around 30 ms is present and a heavy tail is found with processing time growing up to 5 s. The distribution reflects the time required by the cache to retrieve the requested content from the database before serving it. Very low latencies can be related to the video being already cached in the server memory; values in [30,300] ms can be related to disk access; finally values larger than 300 ms can be due to congestion in the back-end or to packet loss recovered by lengthy TCP timeout, or to rare content that has to be fetched from some slower storage system. The fact that Mobile-player response require higher processing time can again be explained by the lower popularity of video content. Note also that the prefetching mechanism (see Sec. 2) implemented by PC-player can also speed-up the content retrieval.

6.2 Bitrate ratio

The download bitrate of the video has a fundamental role in defining the quality of the video playback. In fact, if data is not received fast enough, buffer "underrun" events will be suffered, causing the video playback to pause. To measure the smoothness of the playback, we define the *bitrate ratio* as the ratio between the average session download bitrate and the video encoding bitrate. The first corresponds to the total amount of bytes downloaded aggregating flows of the same video session, divided by the time between the first and the last video packet. According to this definition, a bitrate ratio smaller than 1 is a clear sign of impaired performance.

Fig. 20 reports the fraction of sessions with a bitrate ratio lower than one. Some interesting observation holds: First, the access technology has a clear impact on the performance with the ADSL networks performing worst for more than 10% of the downloads respect to the other networks. Compare indeed EU1-ADSL and EU1-FTTH (the latter offers 10Mb/s full duplex access capacity). Both refer to customer of the same ISP in the same city. Still, EU1-ADSL customers suffer worse performance. Unexpectedly, EU1-Campus performs also

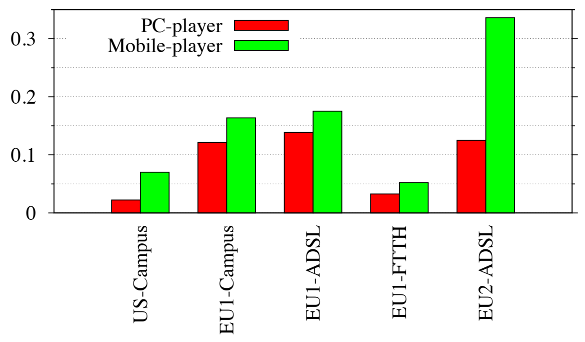


Figure 20: Fraction session with bitrate ratio < 1.

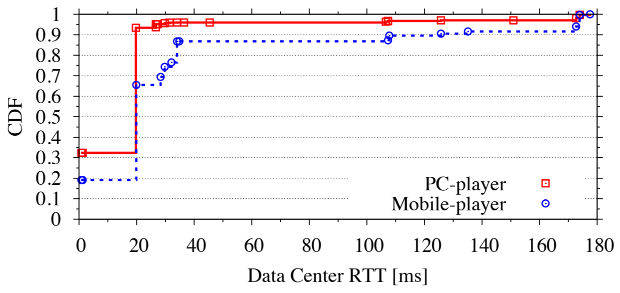


Figure 21: CDF of the fraction of bytes delivered by each data center. EU2-ADSL data set.

quite bad. Further investigation revealed that this is the result of a local network policy that limits the bandwidth of subnets of some dorms. Most of sessions having poor performance are indeed coming from that subnets. Fig. 20 shows that Mobile-player presents consistently lower performance than PC-player. This can be due to the presence of a WiFi network that is used by Mobile-player devices. The shared WiFi connection can indeed impair the download throughput. This is the case for US-Campus.

Other causes of reduced performance can be related to the YouTube infrastructure performing less when serving Mobile-player requests. Consider EU2-ADSL, in which more than 32% of Mobile-player session are performing poorly versus less than 13% of PC-player sessions. We pinpoint that Mobile-player impaired performance are related to the YouTube system. Consider Fig. 21. It reports the CDF of the fraction of bytes downloaded by different video servers respect to the RTT to the EU2-ADSL vantage point. Each point in the figure aggregates video servers that belong to the same CDN data center [13]. We found that EU2-ADSL clients can use a data center which is very close to the vantage point (RTT < 1 ms). However, it can only serve 35% of the PC-player sessions. The majority of sessions are indeed served by a second data center which is 20 ms far from the vantage point. For PC-player, these two data centers handle 96% of video requests. However, due to the lower popularity of Mobile-player accessed content, 35% of Mobile-player sessions are served by other data centers, 10% of which are found outside Europe and suffer RTT > 106 ms. These sessions are impaired by network congestion and exhibit lower download bitrate. Finally, recall the Mobile-player chunking mechanism. The cost of opening a new TCP connection to request a new chunk becomes significant when the RTT is order of hundreds of ms. This impairs the download bitrate too.

Overall, measurements presented in this section show that Mobile-player performance result generally less efficient than PC-player. The intrinsically smaller popularity of Mobile-player accessed videos poses additional challenges to the YouTube CDN infrastructure, which results highly optimized for PC-player as of now.

7. RELATED WORK

We consider three categories of works related to us.

YouTube Videos Characterization: These works have focused on characterizing various aspects of YouTube videos as well the usage patterns. On the one hand, [7] and [17] characterized video popularity, durations, size and playback bitrate, as well as usage pattern statistics such as day versus night accesses and volume of traffic considering a campus network. On the other hand, [3] and [4] crawled the YouTube site for an extended period of time and performed video popularity and user behavior analysis. Further, [3] compares YouTube to other video providers such as Netflix and [4] investigates social networking in YouTube videos. In contrast, our work is focused on the comparison between PC-player and Mobile-player downloads and goes deeper in the characterization of the content also taking advantage of the heterogeneous set of users and networks monitored.

YouTube Infrastructure Studies: These works characterize the YouTube video delivery infrastructure [1, 11, 13]. [11] shows that most YouTube videos are distributed from a single datacenter in the US. [1] shows that a few datacenters in the US were in charge of distributing the videos around the world. Finally, [13] shows that datacenters spread around the world, are in charge of distributing the video and that latency between clients and servers plays a role in content server selection, In contrast, our work is focused on understanding the difference on video delivery between mobile devices and PCs. In particular, we show how mobile devices control the video download rate while in the case of PC-player, the download rate is controlled by the server [2].

User Behavior on Mobile Devices: More recently, there have been several works characterizing high level usage patterns of mobile devices [9, 6, 12]. [9] shows that the number of mobile devices doubled between 2009 and 2010 and that more than 80% of mobile devices traffic is HTTP, with multimedia traffic alone accounting for more than 30% of HTTP. [6] compares the content and flow characteristics of mobile devices and PCs traffic. Using a DPI tool, the authors are able to show that YouTube alone accounts for more than 35% of the Internet traffic. In contrast to these works, we go much deeper into the behavior and performance of users accessing YouTube from mobile devices. In addition, we highlight problems caused by the YouTube infrastructure when delivering videos to mobile devices.

8. CONCLUSIONS

Considering a large and heterogeneous data set of YouTube traces, we have presented our findings about user behavior when watching videos and how the type of user device and infrastructure influence the performance of the playback.

Interestingly, users access YouTube in a very similar manner, independent of their location, the device they use, and the access network that connects them. In addition, they typically watch only a small portion of the video, and typically stick to default player configurations.

While YouTube guarantees very good playback quality by means of aggressive buffering policies, we pinpointed sources of unnecessary data transfer, and the potential for future performance optimization. For example, a less aggressive buffering mechanism could be used to limit the amount of unnecessary traffic when the user aborts the playback. For mobile devices, beside the adoption of the prefetching scheme that is useful to speed-up the video playback, a more precise control of the buffering is essential to avoid duplicate transmission of data. Finally, CDN caching schemes can be improved by leveraging the fact that only a fraction of videos are actually watched by users.

9. REFERENCES

[1] V. K. Adhikari, S. Jain, and Z.-L. Zhang. YouTube Traffic Dynamics and Its Interplay With a Tier-1 ISP: an ISP Perspective. In *ACM IMC*, 2010.

[2] S. Alcock and R. Nelson. Application Flow Control in YouTube Video Streams. *SIGCOMM Comput. Commun. Rev.*, 41:24–30, April 2011.

[3] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing The World's Largest User Generated Content Video System. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, pages 1–14, New York, NY, USA, 2007. ACM.

[4] X. Cheng, C. Dale, and J. Liu. Statistics and Social Network of YouTube Videos. In *IWQoS*, pages 229–238, 2008.

[5] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi. Live Traffic Monitoring with Tstat: Capabilities and Experiences. In *WWIC*, pages 290–301, 2010.

[6] A. Gember, A. Anand, and A. Akella. A Comparative Study of Handheld and Non-handheld Traffic in Campus Wi-Fi Networks. In *PAM11*, volume 6579, pages 173–183, Berlin, Heidelberg, 2011.

[7] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube Traffic Characterization: A View From The Edge. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, pages 15–28, New York, NY, USA, 2007. ACM.

[8] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet Inter-domain Traffic. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 75–86, New York, NY, USA, 2010. ACM.

[9] G. Maier, F. Schneider, and A. Feldmann. A First Look at Mobile Hand-held Device Traffic. PAM'10, pages 161–170, Berlin, Heidelberg, 2010.

[10] The Mobile Internet Report. `http://www.morganstanley.com/institutional/techresearch/mobile_internet_report122009.html`.

[11] M. Saxena, U. Sharan, and S. Fahmy. Analyzing Video Services in Web 2.0: a Global Perspective. In *NOSSDAV*, 2008.

[12] M. S. Shafiq, L. Ji, A. Liu, and J. Wang. Characterizing and Modelling Internet Traffic Dynamics of Cellular Devices. In *Sigmetrics*, 2011.

[13] R. Torres, A. Finamore, J. Kim, M. Mellia, M. Munafò, and S. Rao. Dissecting Video Server Selection Strategies in the YouTube CDN. In *IEEE ICDCS*, 2011.

[14] Tstat Home Page. `http://tstat.polito.it`.

[15] YouTube Blog: Mmm mmm good - YouTube videos now served in WebM. `http://youtube-global.blogspot.com/2011/04/mmm-mmm-good-youtube-videos-now-served.html`.

[16] YouTube Press Room, `www.youtube.com/t/press_statistics`.

[17] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications. *Computer Networks*, 53(4):501–514, 2009.