Analysis and modeling of YouTube traffic

Pablo Ameigeiras, Juan J. Ramos-Munoz, Jorge Navarro-Ortiz, Juan M. Lopez-Soler

Research Center on Information and Communications Technologies (CITIC),
University of Granada, Granada, Spain
Corresponding author: Juan M. Lopez-Soler Email: juanma@ugr.es. Telephone: +34 958242303.
Postal address: Department of Signal Theory, Telematics and Communications,
E.T.S.I. Informática y Telecomunicación,
C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

ABSTRACT – YouTube currently accounts for a significant percentage of the Internet's global traffic. Hence, understanding the characteristics of the YouTube traffic generation pattern can provide a significant advantage in predicting user video quality and in enhancing network design. In this paper we present a characterization of the traffic generated by YouTube when accessed from a regular PC. Based on this characterization, a YouTube server traffic generation model is proposed, which, for example, can be easily implemented in simulation tools. The derived characterization and model are based on experimental evaluations of traffic generated by the application layer of YouTube servers. A YouTube server commences the download with an initial burst and later throttles down the generation rate. If the available bandwidth is reduced (e.g., in the presence of network congestion), the server behaves as if the data excess that cannot be transmitted due to the reduced bandwidth were accumulated at a server's buffer, which is later drained if the bandwidth availability is recovered. As we will show, the video clip encoding rate plays a relevant role in determining the traffic generation rate, and therefore, a cumulative density function for the most viewed video clips will be presented. The proposed traffic generation model was implemented in a YouTube emulation server, and the generated synthetic traffic traces were compared to downloads from the original YouTube server. The results show that the relative error between downloads from the emulation server and the original server does not exceed 6% for the 90% of the considered videos.

1. INTRODUCTION

In recent years, the Internet has experienced enormous increases in traffic from social networking media data and video streaming on-demand web-based services due to the increasing interest in user-generated content. Internet videos from sites such as YouTube, Hulu, and Netflix grew to represent about 40% of the consumer Internet traffic in 2010 [1], while the percentage of the Internet's traffic from Peer-to-Peer (P2P) services has declined in the past few years [1]. Among all audio and video sites, YouTube has become the most dominant, being rated as the third-most visited Internet site (according to [2]); additionally, continuing a growing trend, video traffic has reached 52% of the total traffic in mobile networks at the end of 2011 [3]. Therefore, the analysis and characterization of its traffic is of major importance.

YouTube employs the MP4 container for high-definition (HD) clips and uses the Flash Video (FLV) as the default format for the majority of non-HD clips [4]. While users may upload their content in a variety of media formats, YouTube adapts them to the aforementioned formats before posting [5]. YouTube employs the progressive download technique that enables video playback before the content download is completely finished [5]. It also uses the HTTP/TCP platform, which further distinguishes it from traditional media streaming.

The present paper aims at two main objectives: 1) shed light on the YouTube service from the viewpoint of the progressive download traffic generation carried out by YouTube servers accessed from a regular PC; the rationale for it is because of the algorithms and parameters ruling the traffic generation are not publicly available; and 2) propose a YouTube server traffic generation model. As a result, one major

Keywords – YouTube, YouTube Traffic Generation, Progressive Download, YouTube Data Rate, YouTube Application Flow Control, Progressive Download Traffic Generation.

benefit is that YouTube traffic sources can be easily implemented in network simulation tools and experimental test beds to evaluate the service performance and its end-user quality degradation impact, e.g. as done in [6].

In this work, we focus on FLV-based video clips since the default download from regular PCs use an FLV container for more than 92 percent of the videos clips -as it will be shown in section 3.3-. Furthermore, users stick with default player configurations with negligible voluntary change of video resolutions [7].

Our study includes a complete characterization of the progressive download technique used by YouTube servers. The results will show that the server commences a download by transferring an initial burst of 40 seconds of video data at the Internet's maximum available bandwidth and later applies a throttling algorithm that imposes a data rate equal to 1.25 times the video clip encoding rate. For these phases, YouTube applies a minimum size of the initial burst and a minimum transmission data rate of 250 kbps during the throttling phase.

The experimental results will also show that if the available bandwidth is reduced (e.g., in the presence of network congestion), the server behaves as if the data excess that cannot be transmitted due to the limited Internet bandwidth were accumulated at a server buffer of 2 MB, which is later drained as soon as the bandwidth availability is recovered. It was also identified that the video clip encoding rate plays a key role in determining the download data rate. Therefore, the cumulative density function (CDF) for a large number of randomly-chosen video clips is presented and later fitted by an analytical function.

Based on all of the conducted experiments, we propose a YouTube server traffic generation model. This model was validated by implementing it in a YouTube emulation server, and by comparing the generated synthetic traffic traces with downloads from the original YouTube server. We claim that in the worst case 90% of the video clips generated by our model have a relative error that does not exceed 6%.

The rest of the article is organized as follows: section 2 provides an overview of YouTube server and player operation; section 3 provides the experimental framework and a description of the collected sets of traces; section 4 presents the analysis of the experimental results of the main characteristics of YouTube traffic generation; section 5 provides the pseudo-code of the YouTube server traffic generation model, and presents its experimental validation; and, finally, sections 6 and 7 present the related work and main conclusions respectively.

2. YOUTUBE OPERATION OVERVIEW

This section describes the operation of the YouTube service. Figure 1 summarizes the different functionalities of this service, including both the server and the client.

The video clip download process is initiated by the end-user requesting the YouTube web page of the desired video clip. When the web browser receives the YouTube web page, the embedded player initiates the required signaling (see section 2.1) with the media server (selected from a farm of servers) to indicate the video that is to be played. Then, the player starts progressively downloading the video data (at a receiving data rate in the TCP layer R_r), which is stored in a buffer (as described in section 2.2), and later played with a playback rate equal to the video clip encoding rate V_r. The video clip data are encapsulated in an HTTP response, and then, the data feed a TCP stack (at a traffic generation rate in the application layer G_r) with a proprietary algorithm by the media server. Finally, the data are sent by the TCP stack (at a transmission data rate in the TCP layer T_r) to the client over the Internet (with an available bandwidth B).

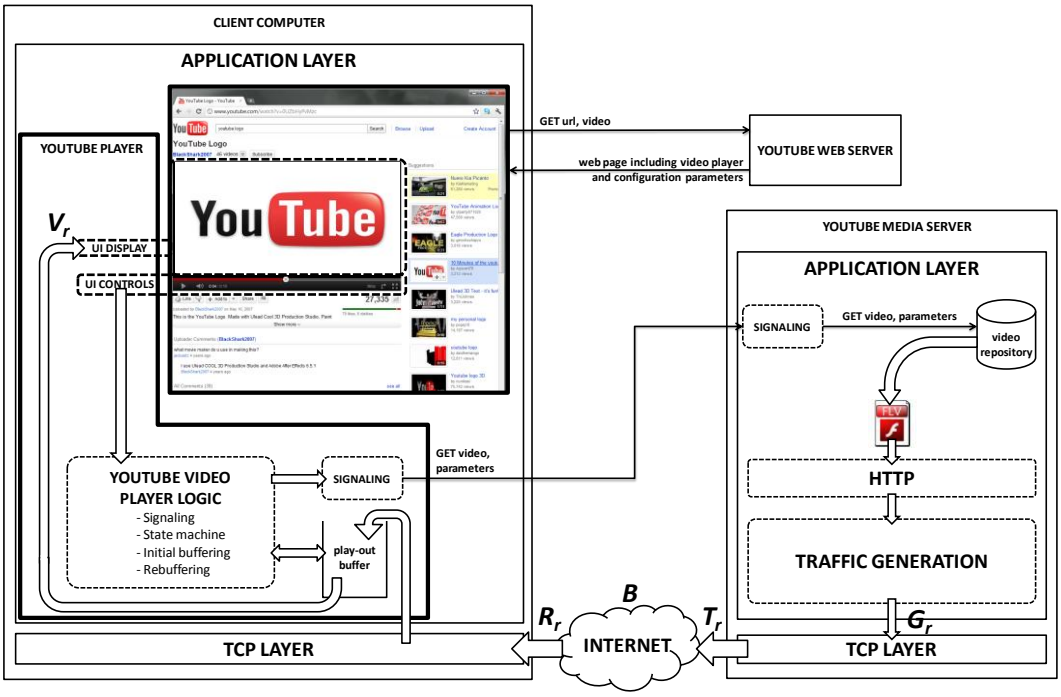


Figure 1. Overview of operation of the YouTube service.

2.1. Signaling

On YouTube, each video has a unique identifier (*video_id*) and can be accessed via a URL at the YouTube site as well as an embedded object in other HTML pages. In the first case, the accessing URL is *http://www.youtube.com/watch?v=<video_id>*. In the second case, the video is accessed as an embeddable object by using the URL *http://www.youtube.com/v/<video_id>*. In both cases, the HTML code contains a customizable Adobe Flash video player [8] provided by YouTube that is downloaded after the web browser parses the page.

The player can be fed with configuration parameters (such as the video clip identifier, video format, and other parameters explained below). When the video clip is embedded in a YouTube web page, the player setup is encoded as JavaScript variables of the HTML page. On the contrary, when the video is embedded in any other web page, the player must obtain the proper parameters by issuing an HTTP request to the URL *http://www.youtube.com/get_video_info?video_id=<video_id>* to obtain the parameters list. The player may choose to change the configuration parameters (for example if it runs in full screen mode) but we will concentrate on the regular case when the parameters are maintained.

After the video player has been configured with the setup parameters, it issues the HTTP request to the streaming server, specified as parameters in the HTML code. Then, a progressive download is performed from that server (called the *media server* hereafter).

We have observed that the video player only issues a single HTTP request to initiate the download and that the video stream is downloaded as an HTTP response with no further client to server signaling. A new HTTP request is required only if the user sets the position for the video playback.

The HTTP request is sent to the URL *http://<media_server>.youtube.com/videoplayback* of the streaming media server along with configuration parameters. These setup parameters, which control the server behavior, are specified as query variables of the service URL. Although to the authors' knowledge the meaning and syntax of the server parameters are not publicly documented, we have identified some of the arguments that govern the download operation. They are summarized in Table 1.

For the majority of non-HD video clips, the HTTP response of the media server encapsulates the requested video and the corresponding audio formatted in an FLV file [9]. The FLV file includes tags that encode the characteristics of the encapsulated media. For instance, the audio tag specifies the audio codec (*SoundFormat*), sampling rate (*SoundRate*) and size of each sample (*SoundSize*). The video tag

includes the codec type (*CodecID* and *VideoData*). This information is essential to decode, render and synchronize the audio and video media as well as to setup the streaming server downloading parameters. The FLV file includes an *onMetaData* tag, which can be accessed from an ActionScript program and describes the video properties (see Table 2).

This information can be accessed easily, and thus, it is reasonable to believe that the media servers may use it to calculate, for instance, the traffic generation rate in the application layer.

Parameter	Description
`sparams`	The list of the parameters included in the request, separated by commas
`id`	A unique video identifier tag
`algorithm`	The algorithm that the media server should use to stream the video; it is fixed to "throttle-factor"
`factor`	Speed factor, expressed as a factor of the video encoding rate; its value is fixed to 1.25
`burst`	The length of video that the server will send for the initial buffering measured in seconds; it is fixed to 40 seconds
`begin`	Playback start time expressed in milliseconds
`itag`	Video format code, equivalent to *fmt* (undocumented URL parameter). FLV-based video clips have *itag* equal to 5 (low quality, 240p), 34 (normal quality, 360p) and 35 (high (non-HD) quality, 480p). See [10] for more information about this parameter.

Table 1. Video request parameters that control progressive download performance.

Parameter	Description
`Duration`	Total duration of the file measured in seconds
`Bytelength`	Total size of the file measured in bytes
`Totaldatarate`	Joint data bit rate in kilobits per second
`Videodatarate`	Video bit rate in kilobits per second
`Audiodatarate`	Audio bit rate in kilobits per second
`Framerate`	Frames per second of the media
`Width, Height`	Width and height of the video frames measured in pixels

Table 2. FLV parameters of the *onMetaData* tag.

2.2. Player operation

As mentioned in section 2.1, the YouTube video player is an Adobe Flash application (in SWF format) that is loaded from the HTML page and obtains its parameters from either JavaScript variables of the HTML page or through an HTTP request. As in other streaming clients, it stores the video data in a play-out buffer that mitigates instantaneous degradation of the network conditions that could affect the end-user's perceived video quality, e.g., causing a playback interruption. These degradations could include fluctuations in the available bandwidth and in the end-to-end delay, as well as packet losses. The player states and the buffering strategy are explained next.

2.3. Player states

The YouTube video player has several states [11], which depend on the current action being performed: *unstarted*, *ended*, *playing*, *paused*, *buffering* and *cued*. These states are represented in Figure 2.

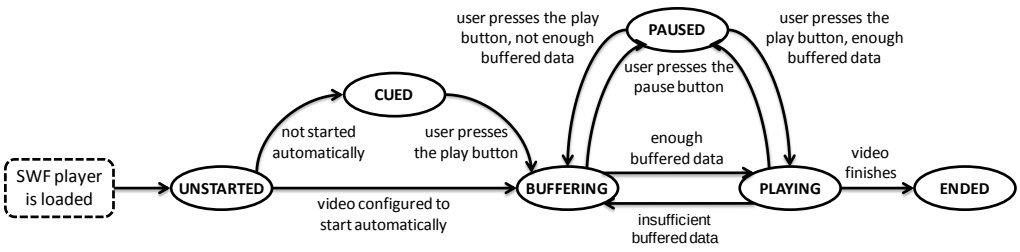


Figure 2. State diagram of the YouTube video player.

When the SWF player is first loaded, it enters the *unstarted* state. Then, the player starts downloading the FLV file, changing its state to *buffering*. Once the player has enough data in its buffer, the playback starts (change to *playing* state). During video play-out the user may pause the video, which causes the player to go to the *paused* state. Assuming that there are no congestion events, the video will continue smoothly, ending without problems, and the player will have entered the *ended* state.

However, network degradations may pause the video playback due to a lack of sufficient buffered data. If this situation occurs, the player will enter the *buffering* state until there are enough data to continue, at which time it will enter the *playing* state. These state transitions may take place several times during short periods, thus causing intermittent and consecutive pauses. In addition, if the player is stopped or configured not to directly play the video when the web page is loaded, it enters the *cued* state (i.e., the video is cued and ready to play). Before the player can enter the *playing* state, a certain amount of data must be stored in the play-out buffer both at the beginning (an initial buffering) and after a playback interruption (a rebuffering event). See [12] for more information about the amount of data stored during a rebuffering event in YouTube.

3. EXPERIMENTAL FRAMEWORK

In this section, we describe the experimental framework used to collect traces of data traffic generated by the YouTube media servers. The framework is composed of a PC connected to a campus network (University of Granada), which in turn is connected to the Internet through the national Academic and Research Network (RedIRIS [13]). During pilot tests, it has been verified that neither the CPU or memory of the PC nor the network connection impeded the normal playback of the video clips. The framework includes three software tools installed in the PC: the Wireshark protocol analyzer [14], a playback monitor and a clip surveyor; the last two were specifically developed for the performed experiments.

3.1. Playback monitor tool

A playback monitor tool has been built with the objective of analyzing the server traffic generation. The tool is based on a Java Servlet and includes a web application that embeds the YouTube player. The tool consists of two parts: 1) a monitoring web page in which the YouTube player is embedded and controlled using the YouTube player API [11] via JavaScript and 2) a Java Servlet that sequentially reads the list of videos to monitor from a configuration file, launches the web browser with our monitoring web page, requests the listed video clips, and later gathers the data generated by the player's API.

For every clip download, the YouTube player API provides -marked with a timestamp- the state of the player, the playback time, and the number of bytes received in the player buffer. This information is dumped periodically every 100 milliseconds. Moreover, the transitions between the player's states are also recorded asynchronously. At the end of the download, the data are gathered by the Servlet and stored in a separate file for post-processing.

The launcher Servlet is configured to intentionally reduce the available bandwidth of the PC's network connection by using SoftPerfect Bandwidth Manager Lite for Windows [15] to introduce controlled bandwidth limitations during the clip download.

3.2. Clip surveyor tool

To characterize the encoding rates of the YouTube video clips, we have also developed a clip surveyor tool, which collects the list of the most viewed videos for a date or range of dates, and downloads the FLV file header for each video clip with the purpose of extracting its main parameters, such as the encoding rate.

By means of the YouTube Data API [11] the tool gathers a list of video clips. To obtain the FLV file header of each clip, the clip surveyor tool automatically launches a web browser (namely Firefox) with the proper URL and stops the download after 30 seconds so that the initial part of the clip is downloaded

but the rest disregarded. The tool FLVTool2 [16] is used to extract the FLV's *onMetaData* information, which contains the video encoding data rate (section 2.1).

3.3. Collected traces

We only consider FLV files for clip characterization because it is the most used file format for video resolutions (more than 92 percent, as exposed below).

Two sets of data traffic traces (sets T1 and T2) have been collected during the weeks between 05/12/2010 and 07/12/2010 in order to analyze the behavior of the YouTube video servers. These trace sets only include the download of the video clips in their default FLV file format.

Two additional sets of data traffic traces (sets T3 and T4) have been collected during the weeks between 26/01/2012 and 27/02/2012 in order to validate the traffic model derived from trace sets T1 and T2. All FLV-based video formats (i.e., *itag* equal to 5, 34 and 35) are considered.

- **Trace set T1**: a set of 95 video clip downloads obtained with the playback monitor tool. This set has been collected to understand the main traffic generation characteristics of the YouTube server: the initial burst (section 4.2), the throttling algorithm (section 4.3), and the chunk size (section 4.4). The video clips have been downloaded with the default quality (89% with *itag*=34 and 11% with *itag*=5), with encoding rates from 140 to 918 kbps (average 510 kbps), and durations from 561 to 659 s (average 605 s). To minimize the effect of network congestion, this trace set was collected at night. Their identifiers (*VideoIDs*) are available at [17].
- **Trace set T2**: a set of 95 video clip downloads (the same set of video clips as in T1) but adding bandwidth limitations. This set has been collected to understand the characteristics of the YouTube server in the presence of bandwidth limitations (section 4.5). This set has been obtained with the playback monitor tool: for every clip, two downloads are performed, reducing the available bandwidth to 50 kbps over time intervals of 15 and 120 seconds, respectively. To minimize the effect of other sources of network congestion on this trace set, these measurements were collected at night.
- **Trace set T3**: a set of 32070 video clips randomly collected using the *random prefix sampling* method presented in [18] that provides an unbiased collection of YouTube video identifiers (*videoID* parameter). Using this sampling method, 12777 *videoIDs* and their available formats (*itag* parameter) were obtained by means of Google Data API queries [19]. The default formats were 34 (FLV format), 18 (MP4 format), and 5 (FLV format) for 92.5%, 7.3% and 0.2% of the considered video identifiers respectively. Concentrating on the available FLV formats of these video identifiers, YouTube provided 12713, 11772, and 7585 video clips with *itags* equal to 5, 34 and 35 respectively, yielding the total set of 32070 video clips. It shall be noticed that none of these *videoID* had *itag* 6, which was an FLV-based format but it is currently inactive. The main information of these video clips was extracted from the metadata of the FLV available containers. These metadata include the audio stream and video stream encoding rates, the encoding rate of the video clip, the video clip duration, the resolution (width and height), the frame rate and the byte length. The 12713 video clips with *itag* =5 (240p) had encoding rates from 38 to 2489 kbps (average 301 kbps) and durations from 1 to 11346s (average 234 s). The 11772 video clips with *itag* = 34 (360p) had encoding rates from 29 to 2333 kbps (average 527 kbps) and durations from 1 to 9131 s (average 242 s). The 7585 video clips with *itag* = 35 (480p) had encoding rates from 33 to 3230 kbps (average 840 kbps) and durations from 1 to 11346 s (average 234 s). These traces have been collected with the clip surveyor tool, and have been used to characterize the video clip encoding rates (see section 4.1). More information about these traces is available at [20].
- **Trace set T4**: a set of 600 video clips has been collected for validation purposes. This set includes 200 video clips for each of the considered formats (*itag* 5, 34 and 35) using the same sampling method as for trace set T3. For each video clip a trace from the real YouTube server and another trace from a customized YouTube emulation server have been obtained. All the traces from the real YouTube server were downloaded over the campus network (University of Granada). Additionally, they were collected at night to minimize the effect of other sources of network congestion. The YouTube emulation server is directly connected through a Local Area Network to our PC. The 200 video clips with *itag* = 5 had encoding rates from 62 to 638 kbps (average 284 kbps) and durations from 120 to 5001 seconds (average 426 seconds). The 200 video clips with *itag* = 34 had encoding rates from 91 to

939 kbps (average 506 kbps) and durations from 121 to 5239 seconds (average 399 seconds). The 200 video clips with *itag* = 35 had encoding rates from 113 to 1371 kbps (average 790 kbps) and durations from 120 to 4538 seconds (average 390 seconds). More information about these traces is available at [21].

4. ANALYSIS OF THE EXPERIMENTAL RESULTS

This section presents the experimental results obtained to evaluate the traffic generation of the application layer of the YouTube server.

4.1. Characterization of the video clip encoding rates and durations

As it will be described in the following subsections, the traffic generation rate of the media server depends on the video clip encoding rate. Hence, we require a characterization of the video clip encoding rates in order to be able to create a model of the traffic that YouTube media servers generate.

To statistically characterize the variety of encoding rates, we extracted the FLV parameters of the *onMetaData* tag from the video clips in trace set T3 and derived the histogram of the encoding rates. As described in section 3.3 we concentrated on the FLV-based video formats (i.e., *itag* equal to 5, 34 and 35).

Because the audio and video codecs used can provide different compression levels depending on the characteristics of the video content, YouTube clips present a wide variety of bit rates. The encoding rates of the audio and video streams have been collected, and their corresponding histograms are shown in Figure 3. Regarding the audio stream, the encoding rates of video clips with *itag* equal to 5 do not exceed 64 kbps, whereas video clips with *itag* equal to 34 and 35 reach up to approximately 128 kbps. Regarding the video stream, the encoding rates strongly differ with the video format. For the *itags* equal to 5, 34 and 35, 99% of the video clips have a video stream encoding rate below 500 kbps, 810 kbps and 1300 kbps respectively.

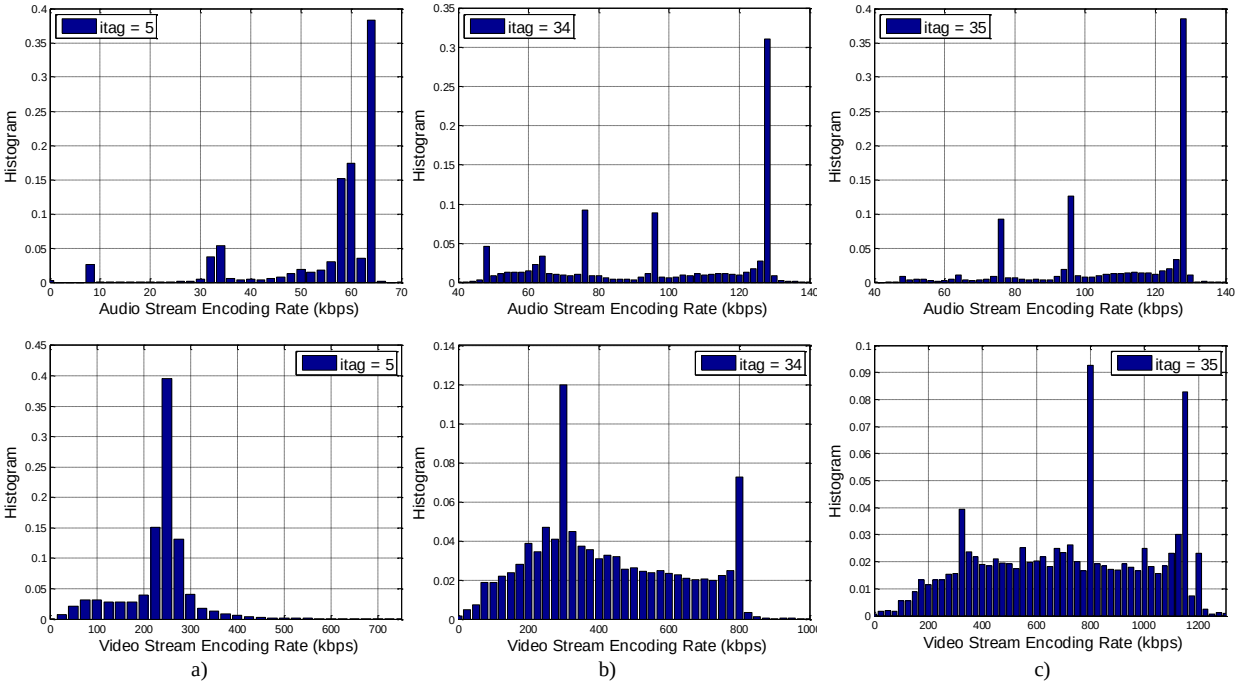


Figure 3. Histogram of the video stream encoding rates of YouTube clips, (a) itag = 5, (b) itag = 34, (c) itag = 35.

For the purpose of statistical characterization, the CDF of the total video clip encoding rate V_r (i.e., the sum of the audio and video streams) and the CDF of the video clip duration d have been computed and are represented in Figure 4. The obtained CDFs have been fitted with the objective of providing

analytical functions that can be implemented in simulation tools. The derived fitting functions, F_V and F_D, are presented in equation (1) and graphically depicted in Figure 4. Please note that the values obtained from the fitting functions shall be bounded between 0 and 1, so the expressions for F_V and F_D are valid if v_r is lower than 650, 930 and 2200 kbps for *itag* 5, 34 and 35 respectively, and d is lower than 5000 s.

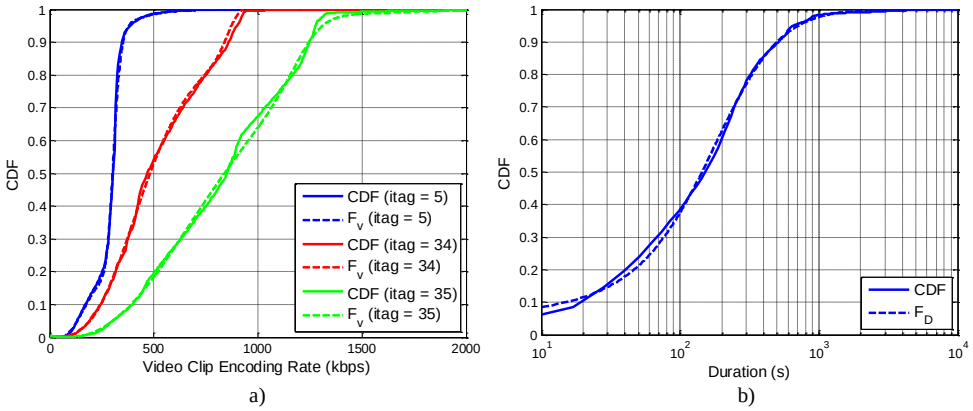


Figure 4. CDFs of (a) the video clip encoding rates and (b) their durations, and their corresponding curve fitting.

$$F_X(itag, x) = \frac{\sum_{i=0}^{5} a_{x,itag,i} x^i}{1 + \sum_{j=0}^{4} b_{x,itag,j} x^{j+1}} \quad (1)$$

where x is the corresponding measurement, i.e. the video clip encoding rate v_r or the video clip duration d, *itag* is 5, 34 or 35, and the corresponding coefficients ($a_{x,itag,i}$ and $b_{x,itag,j}$) are shown in Table 3.

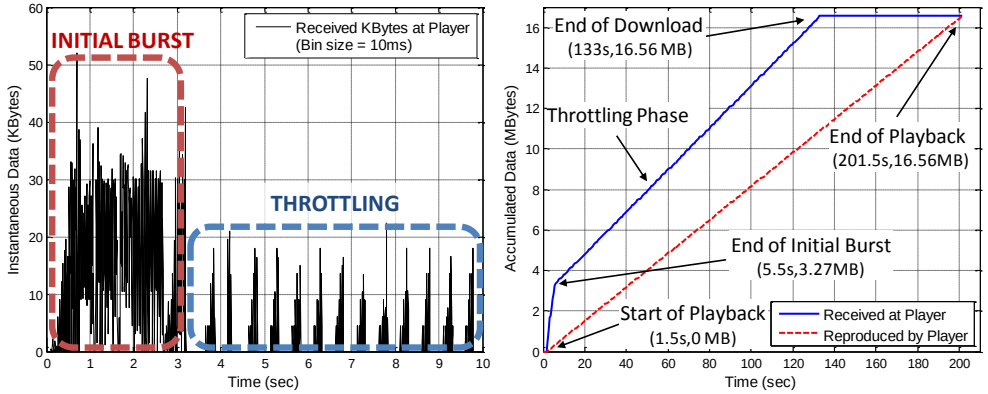
Coefficients	**$x=v_r$ (encoding rate)**			**$x=d$ (duration)**
	***itag*=5**	***itag*=34**	***itag*=35**	**all *itag*s**
$a_{x,itag,0}$	$6.624 \cdot 10^{-3}$	$2.764 \cdot 10^{-3}$	$5.109 \cdot 10^{-3}$	$5.813 \cdot 10^{-2}$
$a_{x,itag,1}$	$-5.530 \cdot 10^{-4}$	$-9.136 \cdot 10^{-5}$	$-1.470 \cdot 10^{-4}$	$2.747 \cdot 10^{-3}$
$a_{x,itag,2}$	$9.850 \cdot 10^{-6}$	$9.675 \cdot 10^{-7}$	$1.057 \cdot 10^{-6}$	$2.082 \cdot 10^{-5}$
$a_{x,itag,3}$	$-5.013 \cdot 10^{-8}$	$1.818 \cdot 10^{-9}$	$-1.422 \cdot 10^{-9}$	0
$a_{x,itag,4}$	$7.926 \cdot 10^{-11}$	$-7.457 \cdot 10^{-12}$	$5.517 \cdot 10^{-13}$	0
$a_{x,itag,5}$	0	$4.935 \cdot 10^{-15}$	0	0
$b_{x,itag,0}$	$-8.908 \cdot 10^{-3}$	$-3.628 \cdot 10^{-3}$	$-2.607 \cdot 10^{-3}$	$2.318 \cdot 10^{-3}$
$b_{x,itag,1}$	$3.579 \cdot 10^{-5}$	$5.834 \cdot 10^{-6}$	$3.423 \cdot 10^{-6}$	$2.088 \cdot 10^{-5}$
$b_{x,itag,2}$	$-8.515 \cdot 10^{-8}$	$-1.431 \cdot 10^{-9}$	$-2.527 \cdot 10^{-9}$	0
$b_{x,itag,3}$	$9.670 \cdot 10^{-11}$	$-6.398 \cdot 10^{-12}$	$8.037 \cdot 10^{-13}$	0
$b_{x,itag,4}$	0	$4.797 \cdot 10^{-15}$	$-2.273 \cdot 10^{-17}$	0

Table 3. Coefficients for the curve fitting of the CDFs of the video clip encoding rates and their durations.

4.2. Initial burst

In this subsection we investigate the operation of the YouTube server during the initial seconds of a progressive download. Based on the information provided by the Playback Monitor Tool, we depict the progressive download of two FLV video clips, which belong to trace set T1, as an example. Figure 5 (a) plots the time evolution of the instantaneous amount of data received by the player at the beginning of the download. Additionally, Figure 5 (b) depicts the time evolution of the accumulated amount of data received in the player's buffer and the accumulated amount of data reproduced by the player during the entire download. The number of bytes reproduced by the player was estimated based on the playback time information and the video clip encoding rate.

As shown in Figure 5, the video clip download commences with a significant burst of data. After this initial burst, the receiving data rate of the client's player is considerably reduced. This effect can be clearly observed in Figure 5 (a), where it can be seen that during the initial few seconds, the amount of data received at the player is significant and later reduced; it can also be clearly observed in Figure 5 (b) by the change in the slope of the accumulated data received at the player after an initial few seconds. Graphical representations of other download examples of trace set T1 are not included here, but they also exhibit the rapid download of an initial burst of data during the beginning of the download. The authors in [7], [22] and [23] also observed that the YouTube server sends the video as fast as possible for an initial buffering period before settling into a constant sending rate.



a) b)

Figure 5. Examples of time evolution of the (a) instantaneous and (b) accumulated received data at the player buffer.

To extend the previous analysis to a large set of video clip downloads, all downloads of trace set T1 have been post-processed in search of an initial burst. This initial burst is identified in each trace by determining the slope change in the accumulated data received by the player between the initial burst and the subsequent phase (hereafter referred to as *throttling phase*). The instantaneous fluctuations of the network bandwidth hindered the identification of the slope change. To mitigate this effect, the accumulated data have been filtered with a 500-ms simple moving average (a 400ms period has also been used and produced nearly identical results). A slope approximation sequence is computed as the difference between consecutive samples of the filtered series. Then, the maximum slope after the initial burst (i.e., during the throttling phase) is computed by considering only the last 20 seconds of the trace.

The observed end of the initial burst is measured as the last instant of the trace when two consecutive samples of the slope approximation sequence surpass the maximum slope of the throttling phase. Table 4 depicts the CDF of the amount of data (measured in seconds of video data, i.e., by dividing the amount of data by the video clip encoding rate) downloaded until the observed end of the initial burst for all downloads of trace set T1. The results show that the majority of the measured sizes amount to approximately 40 seconds of video data. For the remaining downloads, the empirical measurements of their initial bursts slightly differ from 40 seconds, which is caused by short fluctuations in the network's available bandwidth that affect the empirical estimation. However, during the validation process, we detected that for the videos with encoding rates lower than 200 kbps the amount of downloaded data during the initial burst is approximately equal to 40 seconds multiplied by 200kbps.

Size of the initial burst (s)	37	38	39	40	41	42	43	44	45	46	47
Cumulative probability (%)	1.2	1.2	3.6	69.9	89.2	91.6	94.0	97.6	97.6	98.8	100

Table 4. CDF of the initial burst size measured in seconds of video data.

From Table 4 we conclude that in the case of YouTube, there exists an initial burst with a size equivalent to 40 seconds of video content, i.e., a total amount of data equal to 40 multiplied by the video

clip encoding rate assuming a minimum encoding rate of 200kbps. It should be noted that the setup parameter *burst* sent to the HTTP request by the YouTube client is set to 40 seconds (see section 2.1).

The operation of the application layer of the YouTube server at the beginning of the download presented in Figure 5 resembles the Advanced Fast Start of Windows Media Services [24], which sends the first few seconds of data at the maximum available bandwidth of the Internet. The objective of this initial burst is to inject a significant amount of data in the player's buffer. This strategy aims to improve the quality perceived by the end-users beyond that of traditional streaming with no initial burst, in which the player awaits a longer initial buffering delay [25].

4.3. Throttling algorithm

We continue our discussion of the experimental analysis by focusing on the operation of the YouTube server after the initial seconds of a progressive download. As shown in Figure 5 (b), after the initial burst, the slope of the accumulated received data at the player was reduced due to a decrease in the receiving data rate. Let us further analyze this download example. Figure 5 (b) shows that after the initial burst, the slope of the amount of data received in the player's buffer with respect to time (i.e., the receiving data rate) remains approximately constant until the download is completed. This suggests that the application layer in the server throttles down the traffic generation rate, thereby establishing a constant limit on the rate at which the data are fed to the TCP stack during this phase. This throttling effect increases the total time required to complete the file download.

Based on the accumulated data received by the player and assuming an approximately constant transmission data rate during the throttling phase, it is possible to make a simple estimation of the transmission data rate based on Figure 5 (b): T_r = (16.56MB - 3.27MB)/(133s-5.5s) ≈ 875 kbps, where T_r is the transmission data rate. Additionally, the video clip encoding rate V_r = 696 kbps is obtained from the FLV's *onMetaData* information (see section 2.1). Note that V_r could also be estimated based on the accumulated data reproduced by the player from Figure 5 (b) as V_r = (16.56MB)/(201.5s-1.5s) ≈ 695 kbps. Thus, it is interesting to observe that during the throttling phase, the ratio T_r / V_r = 1.256, which indicates that during this phase, the YouTube server progressively downloads the video clip file at a pace approximately 25% faster than it is reproduced by the player. Again, a graphical representation of other examples of downloads of trace set T1 also presents a throttling phase and a similar factor T_r / V_r.

To verify the application of the throttling performed by the YouTube server, all video clip downloads of trace set T1 were also post-processed to obtain the measured download times. Let t_d denote the total time to download a video clip measured in seconds. Then, assuming that the transmission data rate is approximately constant during the throttling phase, t_d can be expressed as

$$t_d = t_{ib} + \frac{8 \cdot s - 40 \cdot V_r}{T_r} \quad (2)$$

where t_{ib} is a variable that represents the time required to download the initial burst, s is the file size measured in bytes, V_r is the video clip encoding rate measured in bps, and T_r is the transmission data rate measured in bps. Let us further assume that $T_r = 1.25 \cdot V_r$ during the throttling phase. Then,

$$t_d = t_{ib} + \frac{8 \cdot s - 40 \cdot V_r}{1.25 \cdot V_r} = t_{ib} + \frac{1}{1.25}(d - 40) \quad (3)$$

where d represents the duration of the video clip measured in seconds (see *duration* parameter available in the *onMetaData* tag in Table 1).

For each video clip, Figure 6 depicts the total download time versus the video clip duration. Figure 6 also represents the linear equation (3), though assuming $t_{ib} = 0$. From the figure, it can be seen that the total download time closely approximates the represented linear equation for video clips longer than 40 seconds and that a small increase is observed due to the variable $t_{ib} > 0$. Again, during the validation

process, we detected that for the videos with encoding rates lower than 200 kbps the transmission data rate during the throttling phase is approximately equal to 250kbps (i.e. 1.25 multiplied by 200kbps).

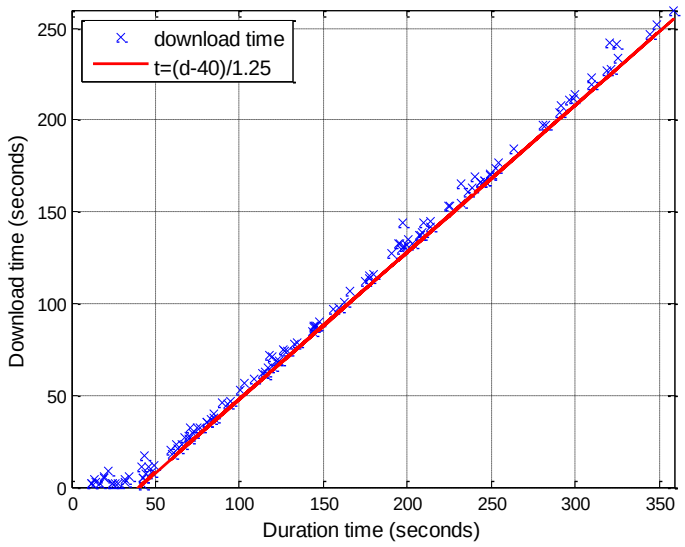


Figure 6. Samples of video clip duration and download time.

From these results, it can be concluded that after the initial burst, the media server throttles down the traffic generation rate, thereby avoiding transferring the data at the maximum available bandwidth. The server sends information at a constant bit rate, and a throttling algorithm (see parameter *algorithm=throttle-factor* in Table 1) is applied that shapes the traffic generation rate according to a throttle factor multiplied by the video clip encoding rate assuming a minimum encoding rate of 200kbps. The so-called throttle factor (see setup parameter *factor* in Table 1) is equal to 1.25. These results are in agreement with [7], [22] and [23].

This throttling procedure is also used in other platforms such as the IIS (Internet Information Services) Media Services delivery platform. It saves bandwidth of media files that might not be played to the end [26]. Additionally, it prevents congestion both at the server and the network because the data transfer is not performed at the Internet's maximum available bandwidth.

4.4. Chunk size

This section analyzes another characteristic of the traffic generated by the YouTube server: during the throttling phase the traffic is generated in chunks of a specific size.

Figure 5 (a) shows the data received instantaneously at the player during the initial seconds of an example download. The figure shows that during the throttling phase, the pattern of reception of data alternates between the reception of data chunks and short periods without packets. To further analyze this characteristic, the instantaneously received data of two additional download examples are depicted in Figure 7. The figures only represent a short time span of the throttling phase during the download. The video clip encoding rates of the selected examples are 135 kbps and 1.089 Mbps, which are close to the lower and the upper limits of the video clip encoding rates, respectively (see section 4.1). The figure clearly shows that in both downloads, the data are received in chunks with a nearly constant period. Further analysis of these two examples reveals that the aggregate payload of the TCP packets grouped in each chunk is exactly equal to 64 KBytes. Moreover, the period between chunks is approximately 64KB/(1.25· V_r), i.e., 64KB divided by the transmission data rate during the throttling phase. This characteristic of the traffic generation can be easily recognized when analyzing a short time span during the throttling phase, as in Figure 7. However, it cannot be identified during the initial burst.

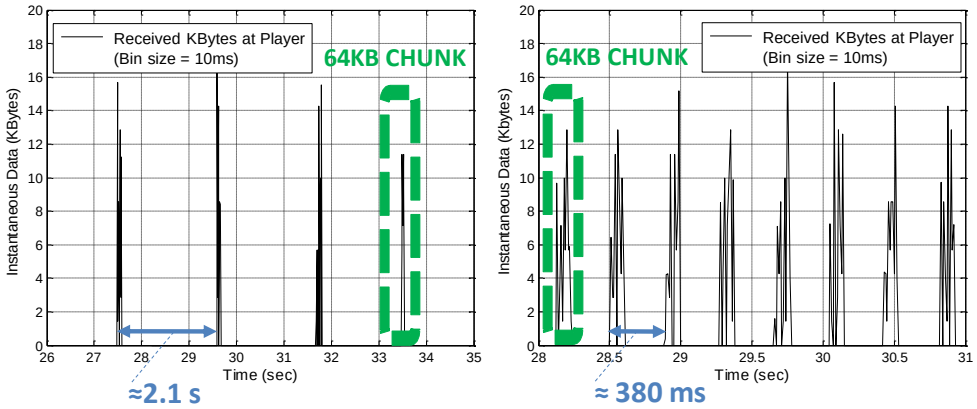


Figure 7. An example of time evolution of the instantaneously received data at the player buffer for video clips with encoding rates of (a) 135 kbps and (b) 1.089 Mbps.

To verify this characteristic of the YouTube server, again all video clip downloads of trace set T1 were also post-processed. In this case, we use the WireShark information to collect the time instants at which the packets arrived at the client computer. Post-processing eliminates the initial bursts of each download. Additionally, the post-processing groups packets into chunks so that two consecutive packets belong to the same chunk if the difference between their arrival times does not exceed a given time threshold. If the difference is longer than the time threshold, the two consecutive packets are assumed to belong to different chunks. Thus, the size of a chunk can be calculated simply by aggregating the size of the payloads of all of its TCP packets. The time threshold used to decide if two consecutive packets belong to the same chunk is selected to be 200 ms. The selection of this period is based on Figure 7 (b) in which the time between the end of the reception of a chunk and the beginning of the next does not exceed 200 ms, and only 1% of the analyzed video clips have an encoding rate larger than 1 Mbps (roughly similar to the one used for Figure 7 (b)).

From the empirically measured chunk sizes, we observe that the majority (96.86%) of the measured chunk sizes are equal to 64 KB. Marginally, chunk sizes in multiples of 64 KB (e.g., 128 KB and 192 KB) were also found due to delay fluctuations in the network that affect the empirical estimation. In particular, just 0.97% of the chunk sizes have 128KB, whereas none of the other sizes exceed the 0.3%. These results are in agreement with [22] and [23] that also found that chunks are typically 64KB in size.

4.5. Effects of available bandwidth reduction

One of the most relevant factors that may impact the performance quality of the YouTube service is the effect of network congestion because it can potentially cause a rebuffering event, which ultimately degrades the video quality perceived by the end-user. Let us assume that a congestion episode takes place at a given instant during the download of a YouTube video clip. Network congestion has several pernicious effects on this flow [27]:

- Increase in the probability of discarding a packet at the network nodes. For the YouTube service, this effect is mitigated by the retransmission capability of TCP, although retransmission implies a longer transmission delay. Additionally, packet discarding also reduces the transmission data rate at the TCP layer.
- Longer queuing delay at the network nodes and, therefore, an increase in end-to-end delay. However, for YouTube flows, this effect is alleviated by the amount of data stored by the player's buffer during the initial burst.
- Reduction in the transmission data rate in the TCP layer, which in turn limits the bandwidth available for the application. If the network congestion episode is long enough, this effect can lead to a rebuffering event.

Of all of the effects of network congestion, the present section concentrates on the influence of the reduction in the available bandwidth caused by a long congestion episode on the operation of the transmitting and receiving entities of the YouTube application layer.

While it is true that the available bandwidth drops in the presence of network congestion, it does not drop at a constant value. However, to conduct controlled experiments, the bandwidth will be limited to a fixed value in the tests discussed in the following subsections.

4.5.1. The influence of network congestion episodes on the playback

During a congestion episode, the transmission data rate in the TCP layer is reduced. If this rate is lower than the play-out rate, the player drains data from the buffer more rapidly than it is received from the Internet, and therefore, the amount of data stored in the player's buffer starts decreasing. An example of this situation is depicted in Figure 8, where it can be seen that at approximately 150 seconds, the player starts receiving data more slowly due to an episode of network congestion; therefore, the amount of data stored in the player's buffer begins to decrease.

Under this circumstance, two possible cases can be envisaged: i) the network congestion episode is short, and the bandwidth availability is recovered before completely emptying the player's buffer; in this case, the rebuffering event is avoided. ii) The network congestion lasts long enough so that the player's buffer eventually runs out of data; in this case, the play-out is paused, and the player starts rebuffering the data. Video clip playback is stopped for the amount of time required to accumulate sufficient data to resume the play-out. This situation is also represented in Figure 8, where it can be seen that the amount of data stored in the player's buffer approaches zero at approximately 215 seconds, leading to the rebuffering event. From the figure, it can also be concluded that a congestion episode is more likely to cause a rebuffering event if it occurs during the early stages of the download because there are fewer data stored in the player's buffer.

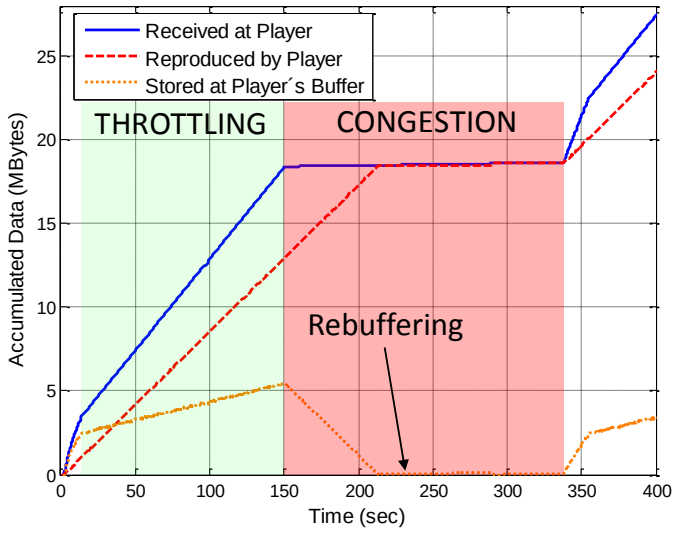


Figure 8. The effect of network congestion on the playback.

4.5.2. Server response to a network congestion episode

A network congestion episode may not only affect the player and the playback of the video clip, but, as it will be described in this section, it may also affect the traffic generation rate of the server's application layer.

To analyze this response, the playback monitor tool presented in 3.1 has been used to download a video clip in which the bandwidth of the network connection of our host machine was intentionally reduced to 50 kbps during a given time interval. Two time intervals are considered: 15 and 120 seconds. The selection of these congestion durations is intentionally chosen to manifest the desired effect. The

selected video clip has an encoding rate of 684 kbps. The experiment has been repeated five times for each time interval. The results obtained after processing the corresponding traces are presented in Figure 9.

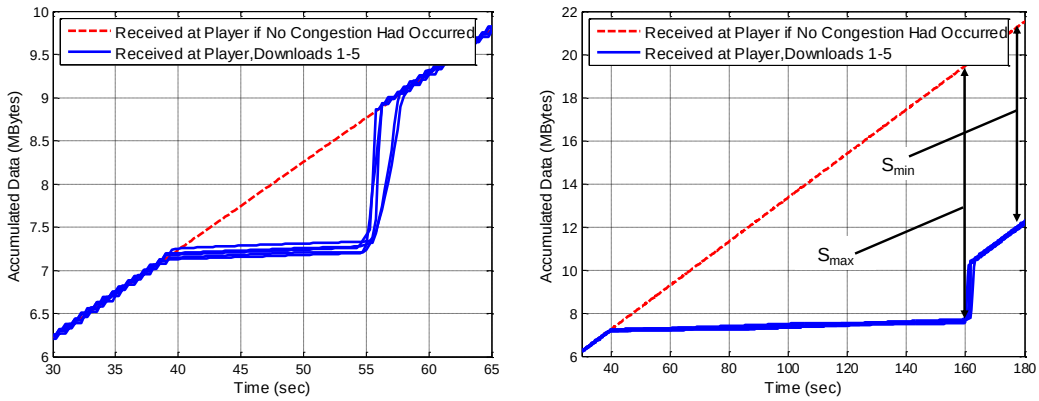


Figure 9. Five video clip downloads with a congestion episode lasting (a) 15 s and (b) 120 s.

In the five downloads represented in Figure 9 (a), it can be seen that after the recovery phase (at time instant 58 seconds and beyond), the amount of data received by the player is the same as if there had been no congestion episode. On the contrary, in none of the five downloads represented in Figure 9 (b) does the amount of data received at the player reach a value that would suggest that no congestion episode had occurred.

In the download examples shown in Figure 9 (a) and (b), the YouTube server acts as if the amount of data that could not be transmitted during the network congestion episode was stored in a buffer. When the episode ended, the server's application layer released the data stored in the buffer at the available bandwidth, which explains the rapid download during the recovery phase. From Figure 9 (a), it can be inferred that the buffer is continuously fed (even during the congestion episode) at the rate indicated by the throttling algorithm, which explains why after the recovery phase the amount of data received by the player is the same as if there had been no congestion episode. However, in the case of Figure 9 (b), the amount of data received by the player does not reach the value that suggests no congestion episode had occurred. This might be caused by a limited buffer size such that if the congestion episode lasts long enough, the buffer is filled up and the filling procedure is blocked. As a consequence, in the case of Figure 9 (b), the time required to complete the download of the video clip is observed to have increased.

To verify the behavior described above, the previous experiment has been repeated for all downloads of trace set T2. For every clip, two downloads are performed, again reducing the bandwidth of the network connection to 50 kbps over time intervals of 15 and 120 seconds, respectively. For every clip, Figure 10 depicts two metrics: i) the measured time required to complete the download and ii) the estimated time required to complete the download if no congestion occurred. The latter metric is computed from equation (3), where t_{ib} is obtained from the experiment.

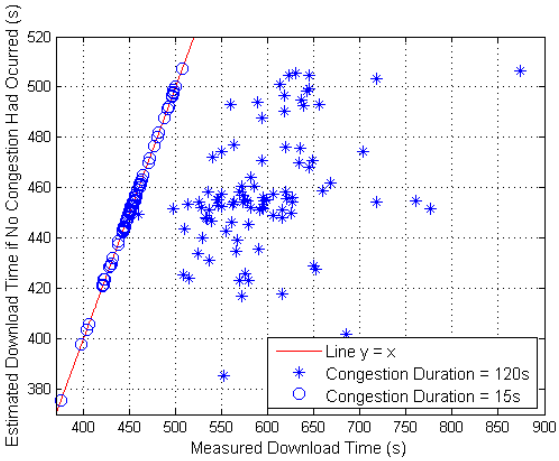


Figure 10. Measured download time versus estimated download time if no congestion occurs. The durations of the congestion episodes are 15 and 120 s.

The results in Figure 10 clearly confirm the behavior observed in Figure 9 for all of the videos of trace set T2. It can be concluded that the duration of the congestion episode determines whether or not the YouTube server buffer can fully compensate the effect of the temporary bandwidth reduction in the total download time. These results suggest that the YouTube server's application layer behaves as a non-greedy source that feeds the TCP stack at the rate imposed by the throttling algorithm. The TCP layer possibly manages and implements the buffering. Then, when a network congestion episode occurs, the transmission data rate at the TCP layer is reduced, and the data excess that cannot be transmitted begins to be stored in the buffer. If the congestion episode is short enough and the transmission data rate at the TCP layer is recovered, and then the buffer is drained. However, when the buffer is full, the application layer is blocked from sending more data to the buffer. However, it is unknown how deep and how long the temporary bandwidth reduction must be to cause a buffer overflow and, therefore, an increase in the total download time.

To determine the size of this YouTube server buffer, we further analyze the traces corresponding to the downloads of trace set T2 with a reduced available bandwidth of 50 kbps that lasts for an interval of 120 seconds. An estimation of the YouTube server buffer is performed for every download based on the results depicted in Figure 9 (b). The estimation is computed as $B_s = S_{max} - S_{min}$, where B_s denotes the server buffer size. S_{max} is computed during the congestion episode and represents the maximum difference between the amount of data received by the player and the amount of data received if no congestion episode occurred. S_{min} is computed after the congestion episode is over and represents the minimum difference between the amount of data received by the player and the amount of data received if no congestion episode occurred. The CDF of the resulting estimations (B_s) of trace set T2 are shown in Table 5. The results show that the estimations approximate a size of 2 MB. Hence, it is assumed that the YouTube server buffer has a maximum size of 2 MB (or equivalently, 32 chunks of size 64 KB each).

B_s (MB)	1.96	1.98	2.00	2.02	2.04	2.06	2.08	2.10
Cumulative probability (%)	3.2	10.6	35.1	61.7	88.3	97.9	98.9	100

Table 5. CDF of the estimated size of the YouTube server buffer B_s.

5. SERVER TRAFFIC GENERATION MODEL

Based on the experimental data presented in previous sections, a simple model of the traffic generated by the YouTube server is proposed by means of an algorithm described in pseudo-code (see Figure 11). The algorithm provides the instants at which the application layer feeds a TCP blocking socket. The socket is to be opened with a sending buffer of 2MB.

The algorithm consists of an initialization block and a filling procedure. At the beginning, the algorithm calculates the initialization parameters, which will be used next. In network simulation tools the

encoding rate and the duration can be obtained by sampling the CDF proposed in equation (1). From these two parameters, the video clip size can be estimated. For the subsequent phases, YouTube assumes a minimum encoding rate equal to 200 kbps. After the initialization of parameters, the server sends an initial burst of data as its pre-catching strategy. Afterwards, the filling procedure writes blocks of 64 KB of data into the TCP socket with a period controlled by the parameter *sending_rate* (which is computed based on throttling factor and the video encoding bit rate). The socket is assumed to transmit the data packets at the pace indicated by the TCP layer. If the TCP socket sending buffer becomes full, the execution of the program is blocked, and the algorithm cannot write more data until the buffer starts to be drained.

5.1. Validation of the proposed YouTube traffic generation model

This section presents the validation of the proposed traffic generation model. For this purpose, we built a server that emulates the YouTube server, providing FLV video clip files at the pace specified by the traffic generation model of Figure 11. Then, we evaluated the accuracy of the synthetic traffic generated by our customized server and the one generated by the original YouTube server.

Initialization procedure

```
if (video_encoding_rate < 200kbps) then video_encoding_rate = 200kbps;
remaining_bytes = video_size_bytes – 40*video_encoding_rate/8;
sending_rate = video_encoding_rate *1.25;
Open a TCP blocking socket with a 2.0 MB Sending Buffer;

Send initial_burst at maximum available bandwidth;
```

Filling procedure

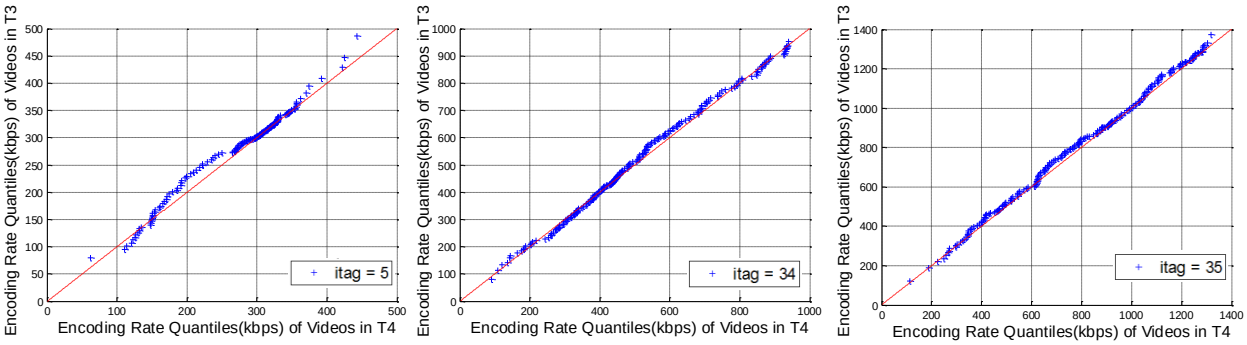
```
While (remaining_bytes>0)
  Write 64 KB into TCP socket;
  remaining_bytes = remaining_bytes – 64 KB;
  Sleep 64 KB/sending_rate seconds;
Endwhile;
```

Figure 11. Pseudo-code of server traffic generation model.

The experimental test bed used for this validation was designed to collect data traffic traces of downloads from the original YouTube server as well as from our YouTube emulation server. For the first case, we reused the framework of section 3. For the second, the playback monitor tool (see section 3.1) installed in our PC was enabled to access the YouTube emulation server that was located in another computer within the same Local Area Network (LAN). The emulation server was implemented with the pseudo-code of Figure 11 in Java, and upon request of the web browser of the Playback Monitor Tool, it starts feeding the TCP socket with the FLV contents at the pace indicated by the proposed traffic model. For downloads from the original YouTube server and from the emulated server, the available bandwidth of the PC's network connection was limited to 5 Mbps. Without a common limit, the connection to the emulated server through the LAN was expected to provide a much higher available bandwidth than the connection through the Internet to the original YouTube server. This would cause a bias in the time required to download the initial burst or to recover after a bandwidth limitation period, which would impede the comparison between the download traces from the original YouTube server and from the emulated one.

For the validation, we aimed at using a representative sample of 200 video clips for each FLV-based video format (i.e., *itag* equal to 5, 34 and 35) in trace set T4. For each format we selected (using the *random prefix sampling* method presented in [18]) a set of 200 video clips whose CDF of the encoding rates matched the CDF of the encoding rates of the extensive video clip collection of trace set T3. The distribution of the encoding rates of the selected video clips in trace set T4 is compared to the ones in

trace set T3 in the QQ plots of Figure 12 for each video format. Moreover, for each format, a two-sample Kolmogorov-Smirnov test failed to reject the null hypothesis that the encoding rates of the video clips in trace sets T4 and T3 are from the same continuous distribution at a significance level of 5%.



a) b) c)

Figure 12. Encoding rates quantile-quantile plots of video clips in trace set T3 and T4, (a) itag = 5, (b) itag = 34, (c) itag = 35.

For every video clip and server, three cases were analyzed: a) no additional bandwidth limitation, b) a 50 kbps additional bandwidth limitation lasting 15 seconds, and c) a 50 kbps additional bandwidth limitation lasting 120 seconds. Again, these bandwidth limitations were performed using SoftPerfect Bandwidth Manager Lite for Windows [15]. For every download, the playback monitor tool provided the amount of accumulated data received by the player's buffer as a function of time.

The download traces from the original YouTube server and from the emulated server have been compared for every video clip and considered case. For the comparison, the instantaneous relative error of the accumulated amount of data has been computed at every sampling instant n as:

$$\varepsilon[n] = \frac{\left|\hat{A}[n] - A[n]\right|}{A[n]} \tag{4}$$

where $\varepsilon[n]$ denotes the instantaneous relative error, $A[n]$ represents the amount of accumulated data received by the player's buffer in the case of the download from the original server, and $\hat{A}[n]$ represents the amount of accumulated data from the emulated server. It should be noted from section 3.1 that our playback monitor tool dumps the collected data every 100 ms, which therefore fixes the period between consecutive samples of the discrete-time sequences $A[n]$ and $\hat{A}[n]$. Finally, for every video clip and considered case, the 90th percentile of the discrete-time sequence $\varepsilon[n]$ has been computed and denoted as $\hat{\varepsilon}$. Figure 13 represents the CDF of $\hat{\varepsilon}$ for the three considered cases.

The results show that in the case of no additional bandwidth limitation 90% of the video clips have a relative error $\hat{\varepsilon}$ that does not exceed 4.5%. In the cases with bandwidth limitation, 90% of the video clips have a relative error $\hat{\varepsilon}$ that does not exceed 6%. Most of the observed error is an artifact of the technique used in our testbed to deal with the variable available bandwidth at Internet. Bandwidth fluctuations at Internet cannot be fully eliminated by the bandwidth limitations introduced to our PC's network connection, which directly affect the relative error of the accumulated amount of data.

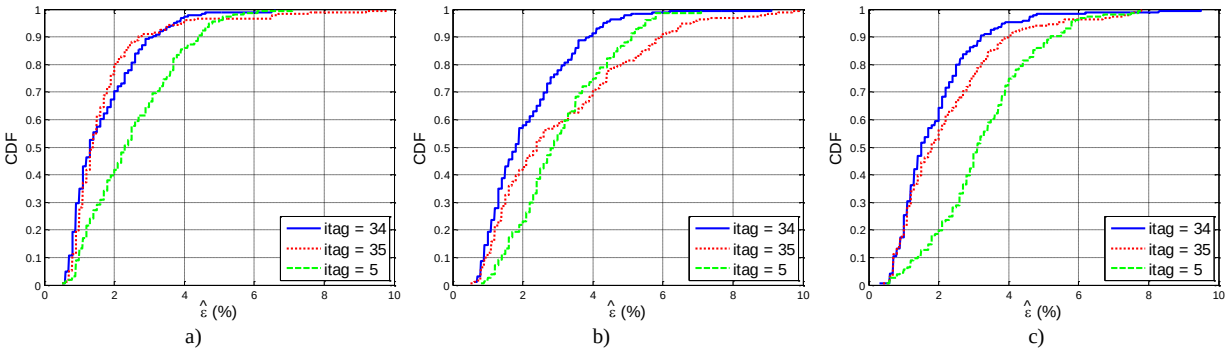


Figure 13. CDF of the relative error $\hat{\varepsilon}$ between downloads from the original YouTube server and from the emulated one for the following cases: (a) no additional bandwidth limitation, (b) a 50 kbps additional bandwidth limitation lasting 15 seconds, and (c) a 50 kbps additional bandwidth limitation lasting 120 seconds

6. RELATED WORK

The impact of new and existing services on traffic volume in current and future backbone networks has been studied in [28]. In this respect, YouTube is the leading video download site, being rated as the third-most visited Internet site [2].

Various studies have been performed to characterize YouTube traffic, each of which has focused on different aspects. Works [5] [29] [30] analyzed relevant characteristics of YouTube videos as their popularity, duration, file size, bit rate, and social network characteristics.

From a different perspective, the work in [31] presented the characteristics of YouTube traffic over a campus network, providing statistics regarding the video request rate over time, the number of requests per client, the session duration and file size, and the authors further analyzed the implications on caching methods.

In [32], the characterization was performed from a DSL provider perspective, indicating that the YouTube traffic characteristics differ significantly from those of other Web traffic and that YouTube servers apply a data rate limitation of about 1.25 Mbps. Similarly, the work in [33] investigated the characteristics of network traffic flows of video-sharing services from a network provider standpoint.

In addition to our work, other studies have focused on the traffic characteristics of YouTube video streaming servers. Alcock and Nelson [22] found some of its basic properties. In particular, they illustrated that it is a standard practice for YouTube servers to send consistently sized blocks of 64KB data at a reduced rate to limit the amount of data that is sent to the client. They also described that during the initial burst phase the amount of data sent by the YouTube server is related to the transmission rate during the throttling phase.

Rao et al. [23] made an extensive description of the streaming strategy of YouTube and Netflix. For YouTube they measured the traffic characteristics for the cases of access via PCs and mobile devices, and additionally considered several containers (Flash, HTML5 and FlashHD). Their measurement results for the case of FLV-file downloads accessed via PC also exhibited an initial burst, and short on-off periods with a throttling factor.

Finamore et al. [7] also compared the traffic generation when accessing YouTube via PCs and mobile devices. They described that mobile devices cannot buffer the entire video so the player progressively requests portions according to the evolution of the playback. Our investigation of YouTube traffic has also found the characteristics presented by Rao [23] and Alcock [22], although our results have shown that YouTube applies a minimum transmission data rate during the throttling phase and a minimum size of the initial burst. Unlike previous works, we have carried out the analysis for the case of reduction of the available bandwidth in the network.

Regarding the modeling of YouTube traffic, Zink et al. [31] proposed a model for evaluating proxy caching schemes that generates user requests that contain video id, client IP, request time, and content size. Additionally, Rao et al. also proposed a mathematical model of the aggregate video streaming. However, to the best of our knowledge, our work is the first to present and validate a model that provides for each video clip the instants and blocks of data at which the YouTube application layer generates the traffic.

7. CONCLUSIONS

This paper characterizes the YouTube service from the viewpoint of traffic generation in the server's application layer, which is very valuable for predicting the video quality perceived by end-users and enhancing network design. The characterization is based on combined information from both YouTube's official documentation and the conducted experiments. The focus has been on FLV-based video clips since, as it has been shown in section 3.3, the default download from regular PCs use an FLV container for more than 92 percent of the videos clips.

The presented results have shown that YouTube's progressive download commences by transferring an initial burst of 40 seconds of data at the Internet's maximum available bandwidth and later applies a throttling algorithm that imposes a data rate equal to 1.25 (i.e., the throttling factor) times the video clip encoding rate. Our results have shown that YouTube applies a minimum transmission data rate of 250 kbps during the throttling phase and a minimum size of the initial burst (equivalent to 40 seconds multiplied by 200 kbps). Moreover, it has also been shown that after the initial burst, the data are sent in chunks of 64 KB.

The YouTube media server reacts to a reduction in the transmission data rate in the TCP layer as if the data excess that cannot be transmitted were accumulated at the server buffer, which is later drained if the bandwidth availability is recovered. The TCP layer possibly manages and implements this server buffer. If the congestion episode is long enough, the server buffer is filled, and the application layer is blocked from sending more data to the buffer. This causes the time required to complete the download of a video clip to increase even if later the bandwidth availability is fully recovered. The experiments conducted have also indicated that the size of the server buffer is approximately 2.0 MB.

Due to the relevance of the video clip encoding rate in determining the download data rate, a CDF of the encoding rates for FLV-file based video clips has been computed. It has been shown that for the *itag* equal to 34 (which is the default format for 92% of the video clips) the video clip encoding rate ranges from 100 kbps to approximately 1 Mbps. This implies that during the throttling phase traffic is generated at a rate that ranges from 250kbps to approximately 1.25Mbps. The obtained CDF of the encoding rates has been fitted by an analytical function. Additionally, the CDF of the video clip duration is also presented and fitted by an analytical function.

Based on all of the conducted experiments, a YouTube server traffic generation model has been formulated, which can be easily implemented in network simulation tools to evaluate service performance and end-user quality.

For validation purposes, we have built a server that emulates the YouTube server, providing FLV video clip files at the pace specified by the proposed traffic generation model. We have compared the accumulated amount of data received by the player when using our customized server and when using the original YouTube server. The results have shown that, for 90% of the considered videos, the relative error does not exceed 4.5% in the case of no additional bandwidth limitation and 6% in the cases of additional bandwidth limitations.

Finally, for future work we are interested in the characterization of the YouTube traffic in the case of network congestion episodes with packet losses and variable bandwidth limitations.

ACKNOWLEDGEMENTS

This work was supported by the "Ministerio de Ciencia e Innovación" of Spain under research project TIN2010-20323. The authors would like to thank the anonymous reviewers for their valuable comments. The authors would also like to thank Professor Jose Carlos Segura-Luna and Jose A. Zamora-Cobo for their very valuable collaboration.

REFERENCES

[1] Cisco Corporation. Cisco Visual Networking Index: forecast and methodology, 2010-2015. White paper. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html. Accessed 25th April 2012.

[2] Alexa Corporation. The Top 500 Sites on the Web. Available: http://www.alexa.com/topsites. Accessed 25th April 2012.

[3] Cisco Corporation. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016. White paper. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf. Accessed 25th April 2012.

[4] Garapati, N. Quality estimation of YouTube video service. Master Thesis. Blekinge Institute of Technology, Sweden. Feb. 2010.

[5] Gill, P., Arlitt, M., Li, Z., and Mahanti, A. YouTube traffic characterization: a view from the edge. *In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. San Diego. 2007. DOI: 10.1145/1298306.1298310

[6] Matera F., Matteotti F., Pasquali P., Rea L., Tarantino A., Baroncini V., Del Prete G., Gaudino G. Comparison between objective and subjective measurements of quality of service over an Optical Wide Area network. *European Transactions on Telecommunications*. Volume 19, Issue 3, pages 233–245, April 2008. DOI: 10.1002/ett.1189.

[7] Finamore A., Mellia M., Munafo M., Rao S. G. YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience, *In Proc. 11th Annual Internet Measurement Conference (IMC '11)*, Berlin, Germany, November 2011.

[8] Adobe Systems Incorporated. Adobe Flash Player. Available: http://www.adobe.com/. Accessed 25th April 2012.

[9] Adobe Systems Incorporated. Video file format specification version 10.1. Available: http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf. Accessed 25th April 2012.

[10] YouTube-DL documentation. Available: http://rg3.github.com/youtube-dl/documentation.html. Accessed 25th April 2012.

[11] Youtube Corporation. YouTube APIs and tools. Available: http://code.google.com/intl/en-US/apis/youtube/overview.html. Accessed 25th April 2012.

[12] Staehle, B., Hirth, M., Pries, R., Wamser, F., and Staehle, D. YoMo: A Youtube application comfort monitoring tool. *In Proceedings of the QoE for Multimedia Content Sharing workshop*, Tampere, Finland. 2010.

[13] Red Iris. Red Iris Weathermap. Available: http://www.rediris.es/conectividad/weathermap/. Accessed 25th April 2012.

[14] Wireshark Corporation. Wireshark network protocol analyzer. Available: http://www.wireshark.org/. Accessed 25th April 2012.

[15] Softperfect Research. SoftPerfect bandwidth manager lite version for Windows. Available: http://www.softperfect.com/products/bandwidth/. Accessed 27th February 2012.

[16] Inlet Media Corporation. FLVTool2 – flash video and meta data manipulation. Available: http://www.inlet-media.de/flvtool2/. Accessed 25th April 2012.

[17] Characterization of trace sets T1 and T2. Available : http://dtstc.ugr.es/tl/downloads/set_t1_t2.csv. Accessed 25th April 2012.

[18] Zhou, J., Li, Y., Adhikari, V.K., and Zhang, Z. Counting YouTube videos via random prefix sampling. *In Proceedings of the 2011 Internet Measurement Conference* (IMC'11), Berlin, Germany. 2011. DOI: 10.1145/2068816.2068851.

[19] Google Data API Protocol – API Query Parameters. Available : http://code.google.com/apis/youtube/2.0/developers_guide_protocol_api_query_parameters.html. Accessed 25th April 2012.

[20] Characterization of trace set T3. Available: http://dtstc.ugr.es/tl/downloads/format_set_t3.xlsx and http://dtstc.ugr.es/tl/downloads/metadata_set_t3.xlsx. Accessed 25th April 2012.

[21] Characterization of trace set T4. Available : http://dtstc.ugr.es/tl/downloads/format_set_t4.xlsx and http://dtstc.ugr.es/tl/downloads/metadata_set_t4.xlsx. Accessed 25th April 2012.

[22] Alcock, S., and Nelson, R. Application flow control in Youtube video streams. *ACM SIGCOMM Computer Communication Review*, vol. 41 no. 2, April 2011. DOI: 10.1145/1971162.1971166

[23] Rao A., Lim Y., Barakat C., Legout A., Towsley D., Dabbous W., Network Characteristics of Video Streaming Traffic. *7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT).* December 2011.

[24] Microsoft Corporation. Windows Media Services features and benefits. Available: http://www.microsoft.com/windows/windowsmedia/forpros/serve/features.aspx. Accessed 27th February 2012.

[25] Varsa, V., and Curcio, I. Transparent end-to-end packet switched streaming service (PSS); RTP usage model (release 9). 3GPP TR 26.937 V9.0.0. 2009.

[26] Microsoft Corporation. IIS Media Services. Available: http://technet.microsoft.com/en-us/library/ee729229(WS.10).aspx. June 10, 2010. Accessed 27th February 2012.

[27] Welzl, M. *Network Congestion Control: Managing Internet Traffic*. John Wiley & Sons. 2005.

[28] Palkopoulou, E., Merkle, C., Schupke, D. A., Gruber, C. G. and Kirstädter, A. Traffic models for future backbone networks -- a service-oriented approach. *European Transactions on Telecommunications.* Volume 22, Issue 4, pages 137–150, June 2011. DOI: 10.1002/ett.1464

[29] Cheng, X., Dale, C., and Liu, J. Understanding the characteristics of Internet short video sharing: YouTube as a case study. Technical Report arXiv:0707.3670v1 [cs.NI], Cornell University, arXiv e-prints. 2007.

[30] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., and Moon, S. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. *In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement.* San Diego. 2007. DOI: 10.1145/1298306.1298309

[31] Zink, M., Suh, K., Gu, Y. and Kurose, J. Characteristics of YouTube network traffic at a campus network – measurements, models, and implications. *Computer Networks* 2008, 53:501–514. DOI: 10.1016/j.comnet.2008.09.022

[32] Plissonneau, L., En-Najjary, T., and Urvoy-Keller G. Revisiting web traffic from a DSL provider perspective: the case of YouTube. *In Proceedings of the 19th ITC Specialist Seminar 2008 on Network Usage and Traffic* (ITC SS 19), Berlin, Germany. 2008.

[33] Mori, T., Kawahara, R., Hasegawa, H., and Shimogawa, S. Characterizing traffic flows originating from large-scale video sharing services. *In Proceedings of Traffic Monitoring and Analysis: Second International Workshop* (TMA 2010). Springer, 2010, 17–31.