YouTube Traffic Detection and Characteristics Extraction

Jorge Navarro-Ortiz, Pablo Ameigeiras, Juan J. Ramos-Munoz,
Jonathan Prados-Garzon, Juan M. Lopez-Soler
Centro de Investigación en Tecnologías de la Información y las Comunicaciones (CITIC),
Universidad de Granada
E.T.S.I. Informática y de Telecomunicación, C/ Periodista Daniel Saucedo Aranda s/n, Granada (Spain)
{jorgenavarro, pameigeiras, jjramos, jpg, juanma}@ugr.es

Abstract- **Video downloading is becoming increasingly relevant in wireless and mobile networks, being more than half of the total traffic according to the latest Cisco global mobile data forecast. In particular, YouTube is the most visited video streaming site. In this paper we propose a method to detect YouTube traffic flows and detect their main characteristics such as resolution and encoding rate. This method would be useful for operators and network administrators since the network would become service aware and e.g. the network could apply service specific policies.**

Keywords- **YouTube, traffic detection, characteristics extraction**

I. INTRODUCTION

The usage of QoS mechanisms which are present in different type of networks, such as wired (e.g. based on the DiffServ or IntServ QoS solutions) and wireless (e.g. IEEE 802.11e and 3G Long Term Evolution), makes necessary to segregate packets into different data flows and obtain their main characteristics. The user experience can be enhanced if the network becomes service aware and the network is able to apply service specific policies.

At present the video services constitute a great part of the traffic in packet switched networks (more than 50 percent of mobile traffic by the end of 2011 and accounting for 70 percent by 2016 [1]). Therefore methods for the detection of YouTube data flows from an aggregate set of packet flows with different services/applications, as well as for the extraction of the main information of the video being downloaded (e.g. resolution, duration and video data rate) will allow network operators to increase the user experience by applying some service aware policies.

The method proposed in this paper is focused on the YouTube video delivery service since it is the most representative video streaming site (3rd in the Alexas global rank [2]). This service is implemented as a video progressive download, i.e. the YouTube client (player) progressively downloads the specified video using the HTTP / TCP protocols while the playback is being performed.

The traffic generated by progressive video download from YouTube media servers [3] is carried over the HTTP protocol. This implies that a simple traffic detector based on the server's port cannot be employed since the HTTP server's port (80) is used mainly for other services, i.e. web browsing.

Existing algorithms to detect YouTube progressive video downloads are based on the statistics of this service. For example, the authors in [4] describe a traffic classifier which is based on flow similarity. Another solution is proposed in [5], using a learning classifier which utilizes logistic regression. In another work, Mori [6] detected YouTube flows based on the IP addresses of YouTube servers. Although this solution is simple and effective, it lacks the possibility of extracting the main characteristics of the downloaded media such as average encoding rate or video duration.

To the best of the authors' knowledge, there is not currently any other solution in the literature using the approach herein described.

The rest of the paper is organized as follows. Section 2 describes the basics of the proposed method, whereas Section 3 depicts the solution in detail. Section 4 presents a sample implementation with license free software and libraries. Section 5 exposes some useful use cases and, finally, Section 6 draws the main conclusions.

II. TRAFFIC DETECTION AND CHARACTERISTICS EXTRACTION

In this paper we propose a method which detects a YouTube traffic flow and extracts its main characteristics. More precisely, this method is able to detect the traffic of the YouTube progressive video download, which is carried over the HTTP protocol.

It shall be noted that our method is not intended for the YouTube video streaming service over RSTP/RTP / UDP protocols, which was previously used for mobile devices. We have checked that current smart phones (based on Android, iOS, Symbian, and Bada) receive the YouTube videos using the HTTP/TCP approach.

Our method for detecting YouTube data flows can be applied to any packet switched network that transports an aggregate set of packet flows from different services/applications. The detection of the YouTube data flow can be used in combination with packet filters to allow the network to segregate YouTube data flows and apply them a service specific treatment with the objective of enhancing both network performance and user's experienced quality.

The basic concept of our proposal is illustrated in Fig. 1. The proposed technique to detect YouTube traffic and its main characteristics, thanks to the inspection of the payloads of certain HTTP packets, provides a mechanism to control and increase the QoE perceived by the end users, and it could be installed in the terminal side or in the network side. The

method could be applied to provide a service specific treatment in order to enhance the network performance as well as the user's experienced quality.

The detection of YouTube flows is based on inspecting the contents of the payloads of the HTTP packets and finding a specific message transferred between a YouTube client and the YouTube server.

In this paper we also propose a method to a priori derive relevant information of the video clip characteristics (e.g. the video clip bit rate). This information can be conveniently exploited by the network during the progressive video download.

In this manner, QoS mechanisms such as Radio Resource Management procedures in wireless networks (e.g. admission control, packet scheduling, load balancing) will be able to discriminate among different data flows and, therefore, will be able to provide traffic differentiation. Moreover, developers / administrators / operators will be able to customize these procedures according to the flows' characteristics.

This procedure can be easily adapted to changes on the YouTube signaling, as long as:

- *For detecting the YouTube traffic*: there shall be a string that uniquely identifies the request for beginning a video progressive download. Currently all YouTube video downloads are initiated by an HTTP request containing the string "GET /videoplayback".

- *For extracting the main video characteristics*: the first packet(s) shall contain a container header with the main video characteristics (e.g. video bit rate, resolution, duration) among its metadata. Currently most YouTube videos are encapsulated onto an FLV container, but this procedure would apply to other containers as long as they have an information header with these metadata.

III. Detailed Technical Description

Our proposal is based on the procedure depicted in Fig. 2 to view a YouTube video based on the HTTP protocol.

Before the user starts viewing a YouTube video, the browser sends an HTTP request to the YouTube web server after clicking on a video link.

The downloaded web page includes the video player (in an SWF container) and the required configuration parameters for the selected video. The player further interchanges signalling messages with the YouTube web server, which finally sends the parameters required to download the video from a YouTube media server (from a farm of servers [3]).

Finally, the player requests the video to the YouTube media server by using an HTTP request. More precisely, the request always starts with the following string:

```
GET /videoplayback?sparams=id
```

which is followed by a number of parameters and their corresponding values.

This TCP packet (hereafter designated *get_videoplayback_packet*) is used to carry this HTTP message and contains the **source IP address**, the **destination IP address**, the **source port** and the **destination port**. Therefore, this packet will be used to obtain these values, which uniquely identifies the data flow used to download the YouTube video.

Most YouTube videos are encapsulated into an FLV container, whose format is specified in [7]. The FLV header contains some tags with information about the video, being some of the main tags:

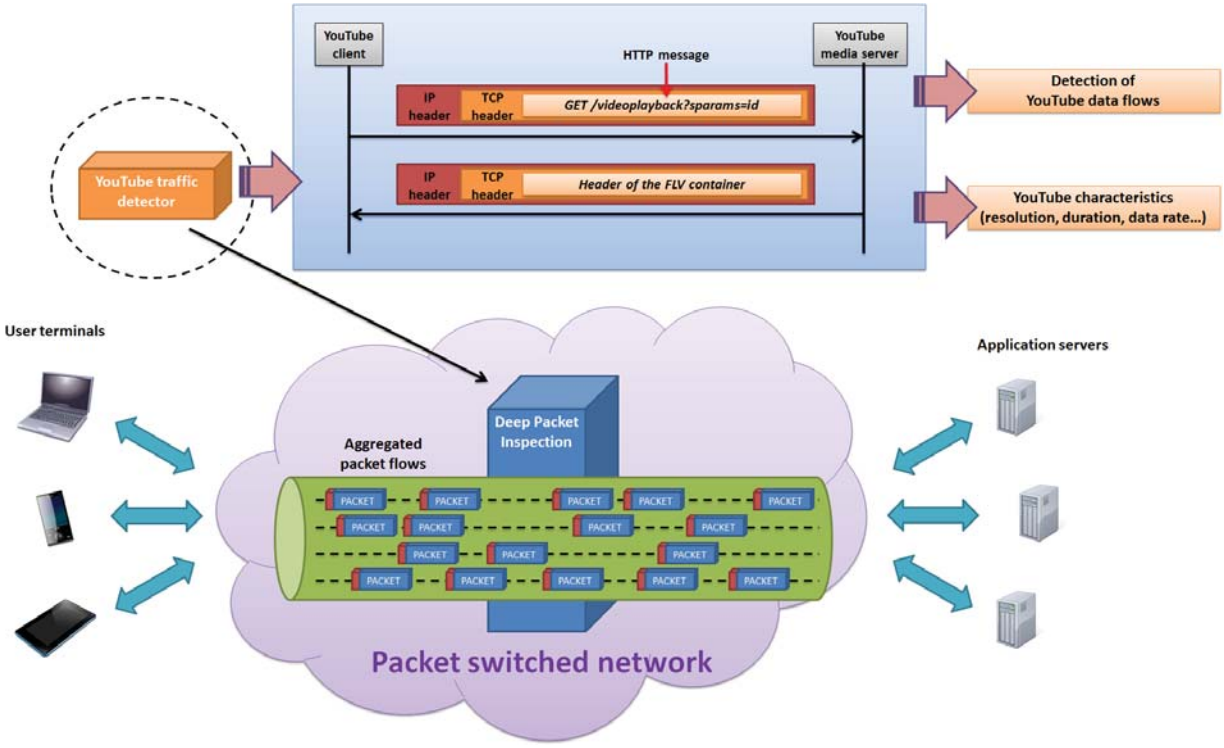

Fig. 1. Basic concept of the YouTube traffic detection method.

- `totalduration`
- `width`
- `height`
- `videodatarate`
- `audiodatarate`
- `totaldatarate`
- `framerate`
- `bytelength`
- `httphostheader`

The first packet from the YouTube media server after the get_videoplayback_packet contains the beginning of the FLV file, i.e., the FLV header. The FLV header starts with a string "FLV" [7] , which can be used to determine whether the video is encapsulated into an FLV container and therefore an FLV header parsing can take place.

If the YouTube video is not encapsulated into an FLV file, then it is encapsulated into an MP4 container [8]. The MP4 container is specified in [9]. The metadata also contains information about the video, e.g. the resolution and the data rate. This information is present in the *moov* atom which can be present in different parts of the video [10]. However, in the case of streaming, the *moov* atom shall be present at the beginning of the video [11] and therefore the methodology used in this paper is also valid for videos in MP4 format.

It is important to note that the string "*videoplayback*" has been searched in packet traces (obtained with *Wireshark* [12]) when viewing videos from the following sites, not being found in any of those traces: hulu.com, metacafe.com, vimeo.com, mtv.com, dailymotion.com, megavideo.com, adobe.com, msn.com, aol.com, myspace.com, yahoo.com, warnerbros.com, disney.com, cbs.com, elmundo.es, elpais.com, 20minutos.es, nbc.com, thetimes.co.uk, tv.tv, citytv.com, spike.com, pandora.tv, muzu.tv, wideo.fr, clarin.com, myvideo.de, wat.tv, kewego.com, brightcove.com, photobucket.com, viddler.com, grindtv.com, liveleak.com, and stupidvideos.com.

IV. Sample Implementation

Our proposal can be implemented by software development, based on a packet sniffing library such as *libpcap* [13]. A sample implementation of YouTube traffic detector is presented in Fig. 3, which is based on the *Python* programming language [14] and the *Scapy* packet manipulation environment [15] . Both *Python* and *Scapy* are available for Windows, Linux, UNIX and MAC OS. *Python* has an OSI approved open source license, and *Scapy* is license free software (GPLv2+). One library available for FLV header parsing (to extract the main characteristics of the flow, e.g. the data rate) is *FLVLib* [16], which is release under a free license (MIT license). The FLV format is specified in [7].

A description of the sample implementation (Fig. 3) is provided next:

- The `main()` function sniffs packets on the chosen network interface, e.g. *eth0*, filtering packets on the TCP port for HTTP (80). If the network interface is not provided as a parameter, the `printUsage()` function prints the syntax for using this application.
- Packets fulfilling the filter criterion are passed to the `callback()` function. If the packet contains data and

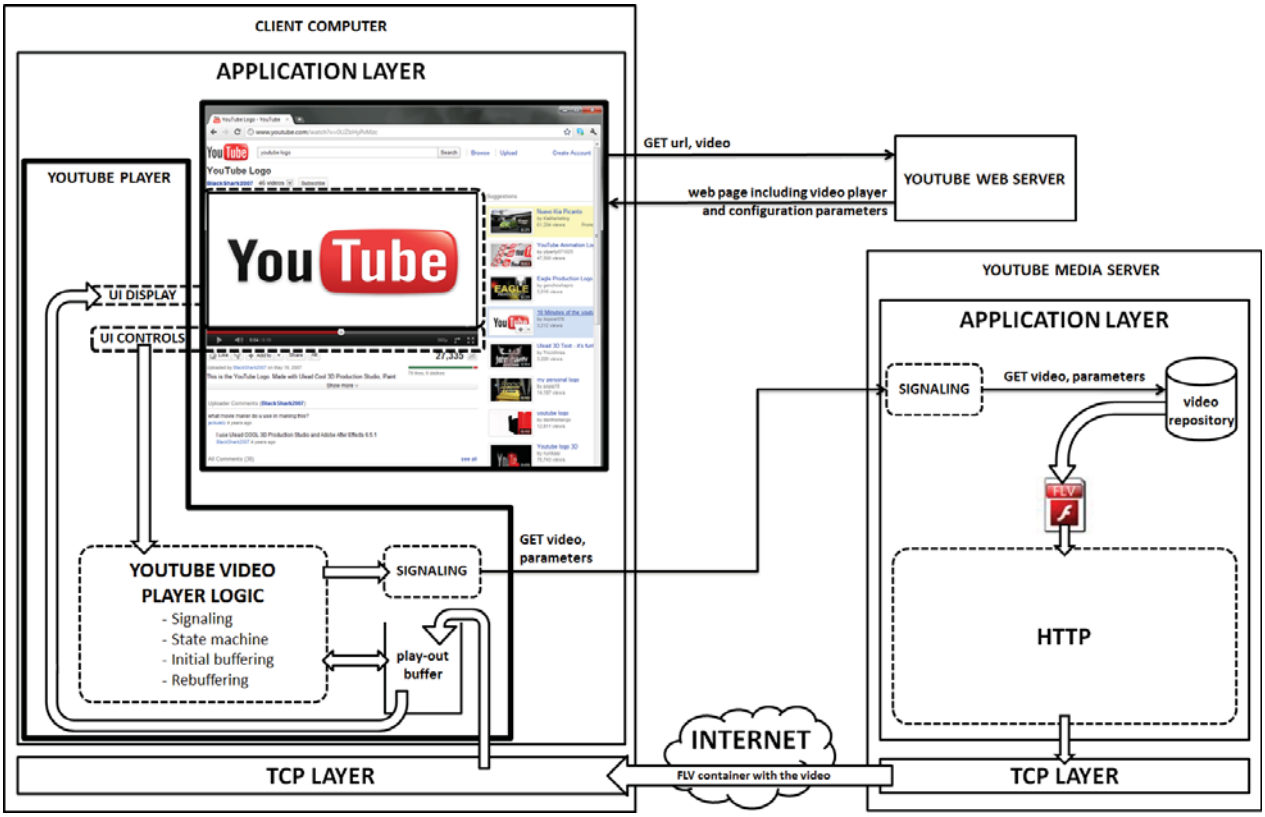

Fig. 2. YouTube video progressive download session.

```
#!/usr/bin/python
#########################################################################
### YouTube server discovery                                          ###
### Copyright (C) 2013 Jorge Navarro-Ortiz et.al.                     ###
### University of Granada                                             ###
#########################################################################

from scapy.all import *
import os
import sys

## Global variables
stringToSearch="'GET /videoplayback?sparams=id" # Request string to the media server

## Initial configuration
conf.verb=0
conf.promisc=0

#####################################
### printUsage: display short help ###
#####################################
def printUsage():
   print "YouTube sniffer."
   print "Usage: " + sys.argv[0] + " <interface>"
   print "E.g.:  " + sys.argv[0] + " eth0"
   print "NOTE: run 'ifaces' on Scapy to check the available interfaces."

#############################################################################################
### checkYouTubeGetVideoPacket: look for the packet sent to the YouTube video server ###
#############################################################################################
def checkYouTubeGetVideoPacket(pkt):
   global stringToSearch

   src=pkt.sprintf("%IP.src%")
   dst=pkt.sprintf("%IP.dst%")
   sport=pkt.sprintf("%IP.sport%")
   dport=pkt.sprintf("%IP.dport%")
   raw=pkt.sprintf("%Raw.load%")

   if raw[0:len(stringToSearch)]==stringToSearch:
      print "IP.src: " + src
      print "IP.dst: " + dst
      print "TCP.sport: " + sport
      print "TCP.dport: " + dport
      print "Raw: " + raw

      nextPacket=sniff(filter="tcp port 80 and host %dst%", count=1)
      FLVHeaderParsingFromPacket(nextPkt)
      exit(0)

#################################################
### callback: called for each packet received ###
#################################################
def callback(pkt):
   sport=pkt.sprintf("%IP.sport%")
   dport=pkt.sprintf("%IP.dport%")
   raw=pkt.sprintf("%Raw.load%")

   if raw!='??':
      if dport == '80':
         checkYouTubeGetVideoPacket(pkt)

############
### main ###
############
def main():
   if (len(sys.argv) < 2):
      printUsage()
      exit(0)

   ## Command line parameters
   conf.iface=sys.argv[1]
   expr='tcp port 80'

   try:
      sniff(filter=expr, prn=callback, store=0)
   except KeyboardInterrupt:
      exit(0)

if __name__ == "__main__":
    main()
```

Fig. 3. YouTube video progressive download session.

the destination port is 80, then the `checkYouTubeGetVideoPacket()` function is called.

- The `checkYouTubeGetVideoPacket()` function checks whether the payload information of the TCP packet matches the string "`GET /videoplayback?sparams=id`". If this is the case, then this is the first packet sent from the YouTube player (in the client device) to the specific YouTube media server (from a farm of servers). The information for detecting this YouTube traffic flow is composed by the source IP address (`src`), the destination IP address (`dst`), the source TCP port (`sport`) and the destination TCP port (`dport`). All these data can be extracted from this TCP packet (see Fig. 4).
- After that, the `sniff()` function is called to obtain the following packet, sent from the YouTube media server to the YouTube player, i.e. the source IP address is `dst`, the destination IP address is `src`, the source TCP port is `dport` and the destination TCP port is `sport`.
- This packet is then analyzed to get the metadata on the FLV header, i.e. for extracting the main video characteristics such as the video data rate, the resolution or the video duration. This analysis is performed in the `FLVHeaderParsingFromPacket()` function. This function is not included for the sake of clarity and readibility. It can be easily implemented with the *FLVLib* [16] library or following the FLV format specifications [7].

V. Examples of Use Cases

One use case for our solution is the detection of YouTube traffic in the Deep Packet Inspector (DPI) functionality in 3G Long Term Evolution (LTE) networks. The Deep Packet Inspection requires creating packet classifiers which, in conjunction with the subscriber's profile, will allow the Policy and Charging Rules Function (PCRF) to initiate the establishment of a dedicated bearer. The parameters of this dedicated bearer (QoS Class Identifier (QCI), Guaranteed Bit Rate (GBR), Maximum Bit Rate (MBR), and Allocation and Retention Priority (ARP)) will be used in the Radio Resource Management (RRM) algorithms to assign the required resources for that specific data flow. Our proposal provides the method to generate such packet classifiers and tune these parameters (e.g. GBR and MBR can be computed considering the flow's data rate) for the case of YouTube traffic. A sample scenario considering this use case is shown in Fig. 5.

A second use case of this solution is the traffic classification when a computer, e.g. a laptop, is connected to a 3G network through a modem. In this situation, the modem's connection software –which is executed at the computer– could implement the traffic classifier proposed in this document, marking the packets (e.g. with the TOS field [17] or the DSCP field [18] of the IP header). In addition, it may signal the flow's data rate to the network for reserving the required resources. Similarly, this packet classification could also be used in Wi-Fi networks supporting the EDCA and/or the HCCA medium access mechanisms [19] for QoS support.

A third use case could be the identification of YouTube streams in order to characterize their traffic profiles in terms of throughput requirements and load generation. This characterization would be very useful for network planning and design but also for resource management mechanisms such as admission control and scheduling.

Another use case is the capability of reducing the bandwidth requirements for video streams in congested networks by transforming the bitrates of video streams. This could be done by transforming the HTTP requests from users' equipment to the YouTube servers so that the computer appears to ask for the same video stream but in lower quality.

This procedure may be also used for the discrimination of YouTube traffic flows when entering a DiffServ network, so the packets belonging to those flows can be marked and treated according to their QoS requirements. Those QoS requirements could be computed from the FLV metadata, e.g. considering the video clip data rate.

Similarly, the detection of YouTube traffic could be used in firewalls in order to block (or boost) this service in specific networks, e.g. in an enterprise environment.

VI. Conclusions

The YouTube video delivery service is highly influenced by network QoS metrics such as delay and throughput, which may cause playback interruptions and therefore negatively impact on the end-user experienced quality. Our proposal provides the means to differentiate YouTube traffic flows from other types of traffic flows, therefore being able to provide a differentiated treatment, e.g. in QoS-related mechanisms / algorithms.

Furthermore, the metadata extracted from the FLV header can be used for collecting video statistics (e.g. resolution, video encoding rate, audio encoding rate, video duration, etcetera) and for tuning QoS mechanisms (e.g. to compute the QoS requirements such as average throughput, average

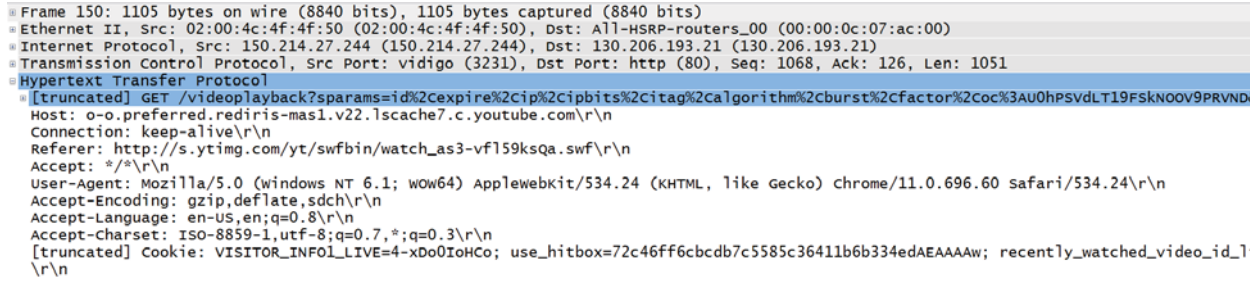

Fig. 4. Sample *get_videoplayback_packet* packet captured with *WireShark*.

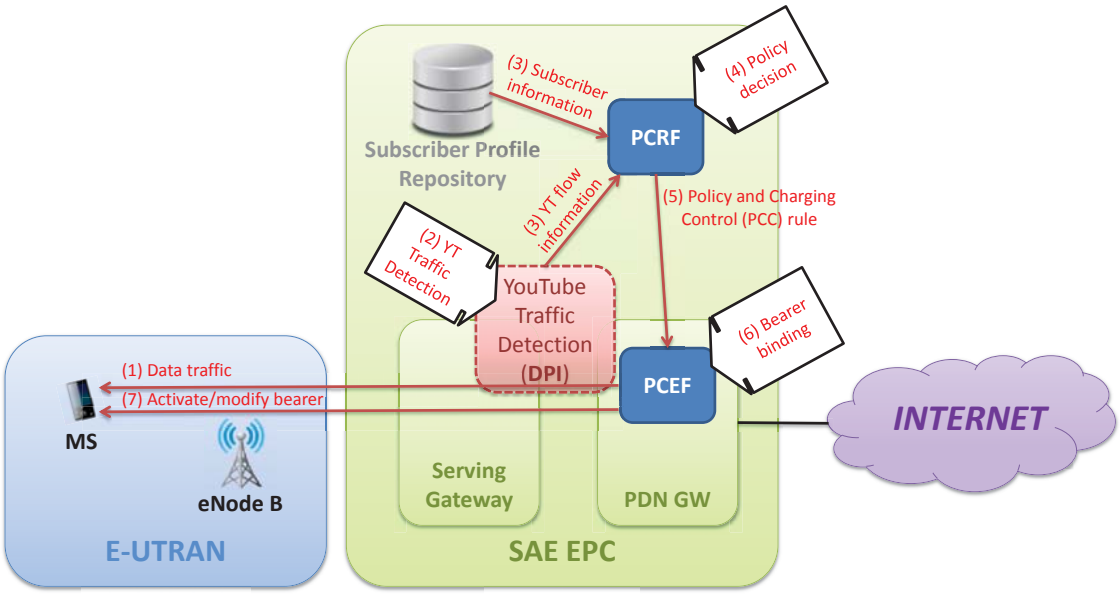

Fig. 5. Sample scenario with an LTE network using the proposed YouTube traffic detector.

packet delay, etcetera).

Both the traffic differentiation and the video characterization can be used by operators and/or network administrators to guarantee the YouTube service requirements.

The technical advantages of the proposed method, compared to other approaches, are:

- This procedure is based on the existing signaling for the YouTube service, not on traffic patterns / statistics. For that reason, it is not possible to have false positives or false negatives when detecting the YouTube traffic. Therefore, our proposal improves the accuracy of the YouTube traffic detection compared to other existing solutions, which are based on traffic patterns / statistics.
- In addition, its implementation is simple but effective, as shown in the sample code included, compared to other existing solutions which require complex statistical computations.
- Moreover, these solutions do not extract information from the video such as resolution, video data rate, duration, etcetera. Our proposal obtains these metadata which could be used for RRM mechanisms such as admission control, resource reservation, resource assignment or for collecting traffic statistics which can be used later for traffic engineering, network planning or troubleshooting.

Acknowledgments

This work was supported by the Ministerio de Ciencia e Innovación of Spain (project TIN2010-20323).

References

[1] Cisco Corporation, Cisco visual networking index: global mobile data traffic forecast update, 2011-2016. white paper. Available: http://www.cisco.com/

[2] Alexa, The Web Information Company. Available: http://www.alexa.com/siteinfo/youtube.com

[3] V. K. Adhikari, S. ain, G. Ranjan, and Z. Zhang, "Understanding data-center driven content distribution", proceedings of the ACM CoNEXT Student Workshop (CoNEXT '10 Student Workshop), New York, USA, December 2010.

[4] J. Y. Chung, B. Park, Y. J. Won, J. Strassner, and J. W. Hong, "Traffic classification based on flow similarity", proceedings of the 9th IEEE International Workshop on IP Operations and Management (IPOM '09), Venice, Italy, October 2009.

[5] T. En-Najjary, M. Pietrzyk, "Application-based feature selection for Internet traffic classification", proceedings of the 22nd International Teletraffic Congress (ITC 2010), Amsterdam, Netherlands, September 2010.

[6] T. Mori, R. Kawahara, H. Hasegawa, S. Shimogawa, "Characterizing Traffic Flows Originating from Large-Scale Video Sharing Services," Proceedings of the Second international conference on Traffic Monitoring and Analysis (TMA'10), Zurich, Switzerland, April 7, 2010.

[7] Adobe Flash Video File format specification, version 10.1, 2010. Available: http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf

[8] P. Ameigeiras, J.J. Ramos-Munoz, J. Navarro-Ortiz, J.M. Lopez-Soler, "Analysis and Modeling of YouTube Traffic," Transactions on Emerging Telecommunications Tecnologies, June 2012.

[9] M. Levkov, "Understanding the MPEG-4 movie atom". Available: http://www.adobe.com/devnet/video/articles/mp4_movie_atom.html

[10] ISO/IEC 14496-14:2003, "Part 14: MP4 file format", 2003.

[11] Android Supported Media Formats. Available: http://developer.android.com/guide/appendix/media-formats.html

[12] Wireshark network protocol analyzer. Available: http://www.wireshark.org/

[13] LibPCap, a portable C++ library for network traffic capture. Available: http://www.tcpdump.org/

[14] Python Programming Language – Official Website. Available: http://www.python.org/

[15] Scapy, an interactive packet manipulation program. Available: http://www.secdev.org/projects/scapy/

[16] FLVLib, a library for parsing, modifying and verifying FLV files. Available: http://wulczer.org/flvlib/

[17] P. Almquist, "RFC 1349: Type of Services in the Internet Protocol suite", July 1992. Available: http://tools.ietf.org/html/rfc1349

[18] K. Nichols, S. Blake, F. Baker, and D. Black, "RFC 2474: Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers", December 1998. Available: http://tools.ietf.org/html/rfc2474

[19] IEEE 802.11-2012, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.