# YOUTUBE VIDEO PACKET LOSS & RETRANSMISSION

**BY**

**M.Ibnul Morshed**

**ID: 103-19-1268**

**Kazi Farhad Robi**

**ID:111-19-1318**

**Md Ripon Miah**

**ID:103-19-1267**

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Electronics and Telecommunication Engineering

**Supervised By**

**Dr. A.K.M. Fazlul Haque**

Professor and Head

Department of ETE

Faculty of Science & Information Technology

Daffodil International University


**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**October , 2014**

# APPROVAL

This thesis titled **"Youtube Video Packet Loss and Retransmission"** has been submitted in partial fulfillment of requirements for the Bachelor of Science degree at Daffodil International University and is submitted by M.Ibnul Morshed, Kazi Farhad Robi and Ripon Miah and its to be made available to borrowers, as are copies of regular theses and dissertations. Brief quotations from this manuscript are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Department of Electronics and Telecommunication when the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author. The presentation has been held 27th September 2014.

**BOARD OF EXAMINERS**

**Dr. A.K.M. Fazlul Haque** — **Chairman**

Professor and Head

Department of ETE

Faculty of Science & Information Technology

Daffodil International University

**Mr.Md.Taslim Arefin** — **Internal Examiner**

Assistant Professor

Department of ETE

Faculty of Science & Information Technology

Daffodil International University

_____

**Mrs.Shahina Haque** **Internal Examiner**

Assistant Professor

Department of ETE

Faculty of Science & Information Technology

Daffodil International University

_____

**Dr. Subrata Kumar Aditya** **External Examiner**

Professor and Chairman

Department of Applied Physics,

Electronics and Communication Engineering

University of Dhaka.

# DECLARATION

This thesis is a presentation of our original research work. Wherever contributions of others without us are involved, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions .The work was done under supervision of Professor **Dr. A.K.M. Fazlul Haque,** Associate Professor & Head, Department Of Electronics and Telecommunication Engineering, Daffodil International University, Dhaka.

**Supervisor's Declaration**

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.:

- the research was carried out and the dissertation was prepared under my direct supervision;
- except where otherwise approved by the Academic Administration Committee of Daffodil international University, the research was conducted in accordance with the degree regulations and house rules;
- the dissertation/thesis (please circle one)represents the original research work of the candidates;
- the contribution made to the research by me, by other members of the supervisory team, by other members of staff of the University and by others was consistent with normal supervisory practice.

**Supervised by:**

_____

**Dr. A.K.M. Fazlul Haque** **Chairman**

Professor and Head

Department of ETE

Faculty of Science & Information Technology

Daffodil International University

**Candidate's Declaration**

We confirm that:

- this dissertation/thesis (please circle one) represents our own work;
- the contribution of any supervisors and others without us to the research and to the dissertation/thesis (please circle one) was consistent with normal supervisory practice.

**Candidates:**

_____

**M.Ibnul Morshed**

**ID: 103-19-1268**

Department Of ETE

Daffodil International University

_____

**Kazi Farhad Robi**

**ID:111-19-1318**

Department Of ETE

Daffodil International University

_____

**Md Ripon Miah**

**ID:103-19-1267**

Department Of ETE

Daffodil International University

## ACKNOWLEDGEMENTS

First and above all, we praise ALLAH, the almighty for providing me this opportunity and granting us the capability to proceed successfully. This thesis appears in its current form due to the assistance and guidance of several people. We would therefore like to offer my sincere thanks to all of them. We fell grateful to express our boundless honor and respect to our supervisor, **Dr. A.K.M. Fazlul Haque**, Associate Professor & Head, Department Of Electronics and Telecommunication Engineering, Daffodil International University, Dhaka and also want to express our deep thanks to him for the trust, the insightful discussion, offering valuable advice, for your support during the whole period of the study, and especially for your patience and guidance during the writing process and for your advices and your friendly assistance with various problems all the time, especially for your help with the paperwork. We would like to thank the members of the reading committee, Assistant Prof. **Taslim Arefin**, Assistant Prof. **Shahina Haque** and **Dr. Subrata Kumar Aditya professor** for their excellent advises and detailed review during the preparation of this thesis. Thanks to all classmate because of yours greatly excellent appreciate assistance and yours spiritual supports for us during our Bachelor study. We will never forget the time that we were together and discussed about all thing.

# ABSTRACT

A video hosting service allows individuals to upload video content to share or place on a website. Users generally will upload via the hosting service's website, mobile or desktop applications or APIs. The type of video content uploaded can be anything from short video clips all the way to full length movies. The video host will then store the video on its server, and show the individual different types of embed codes or links to allow others to view this video. The website, mainly used as the video hosting website, is usually called the video sharing website.

Video sharing services like YouTube have become very popular which consequently results in a drastic shift of the Internet traffic statistic. YouTube is used as concrete example and case study for video delivery over the Internet, since it is not only the most popular online video platform, but also generates a large share of traffic on today's Internet. When transmitting video con-tent over packet based networks, stringent quality of service (QoS) constraints must be met in order to provide the com-parable level of quality to a traditional broadcast television. However, the packet transmission is influenced by delays and losses of data packets which can have devastating influence on the perceived quality of the video. Therefore, we conducted an experimental evaluation of HTTP based video transmission focusing on how they react to packet delay and loss. Through this analysis we investigated how long video playback is stalled and how often re-buffering events take place. Our analysis re-vealed threshold levels for the packet delay, packet losses and network throughput which should not be exceeded in order to preserve smooth video transmission.

# TABLE OF CONTENT

# List of Figure

# List of Table

# Chapter 1

## Introduction

**1.1 Background:** we are trying to create a cmd program for our tests, depicted to Analyses are conducted in two phases. In phase one, test a client computer simulate a streaming application by issuing HTTP5 requests to videos at a YouTube cache. In the web based video data is stored and analyzed for its frame characteristics, such as the size, type and relative playback time. Furthermore, the client captures the packet trace using Command Prompt allowing online analysis of the packets to and from the HTTP server. In second phase we use caps to determine traffic analysis and also to check security analysis and HTTP analysis of youtube video transmission which is done through a network emulation node using of altering the network QoS parameters, such as packet delay distribution, packet loss rate, or transmission throughput. Random packet losses are limited due to access network link layer and transport protocol retransmissions. We focus on asymmetric access network such as ADSL phone lines which are assumed to from the bottleneck links. From this analysis, we can determine how much data is required to play each frame of the video without any delay to allow for an uninterrupted playback. On this basis, we generate statistics about user-perceivable artifacts such as re-buffering events that would occur during the playback. For our experiment we used a 92 s video file encoded at 850 Kbit/ps.

**1.2 Aim of the research work:** During the past years video sharing services like YouTube in the US, Smiley in Japan, the now defunct Megavideo in Hong-Kong, and Dailymotion in France have become very popular. YouTube users alone request millions of videos every day. Consequently, popularity of this kind results in a drastic shift in Internet traffic statistic, which reports that the share of P2P traffic is declining, primarily due to an increase in traffic from Web-based video sharing services [1].So far, there is no indication that this trend will decrease and indeed is more likely to sustain.

Thus, fulfilling the rising demand for video traffic will be a challenging task for both content providers as well as ISPs (Internet Service Providers).Video streaming in the above mentioned services is either web-based or HTTP-based, therefore being transported using the TCP. The TCP is currently the most widely used transport protocol in the Internet but conventionally regarded as inappropriate for media streaming. The primary reason lies in the TCP reliability and retransmission mechanisms which can lead to undesirable transmission delays and may violate timeliness requirements for streamed live media. In this context, coping with packet delay and

loss, which can occur due to congestion or packet corruption, demands new solutions as classical transmission procedures, used in unreliable protocols, e.g. UDP, may be not sufficient. It should also be taken into account that the HTTP and TCP are general purpose protocols and were not specifically designed or optimized for streaming media delivery. Thus, attempts are being made to adapt media delivery to the Internet instead of trying to adapt the Internet to multimedia content streaming. In our work we concentrate on YouTube which represents a service that is unlike the traditional VoD systems in several important aspects. From our perspective, the most important difference between YouTube and other more traditional VoD systems is that the latter usually offer professionally-produced video content such as movies, news, sport events, or TV series. The quality and popularity of this content are well-controlled and predictable. In contrast, YouTube videos can be uploaded by anyone with access to the Internet. The quality of these video clips varies significantly making network optimizations for specific content unreasonable. Internet connections are characterized by a number of statistically determined characteristics including latency and reliability. These traits are not guaranteed – in fact, they
can fluctuate considerably depending on the local ISP network load, remote server load, background traffic, as well as network infrastructure quality. Video delivered by more traditional channels such as satellite, DVD, cable or digital TV broadcasting requires usually not too much buffering space at a client side because data arrives at a media player with mostly deterministic delay, rate and very limited or infrequent data drops. Video delivered over the Internet is much more problematic because there is no guarantee that the data will flow to a user at a sufficient rate and determined delay. Instead, it arrives with a rate and delay that can change consistently during video file transmission. Therefore, buffering is of increasing importance for video streams when they are transmitted over the Internet, including Web-based streaming. The YouTube client software manages buffering and playing of the received content using several behaviors [2].In our work we study the efficiency of three possible streaming client playback strategies. Our goal is to investigate how often the player buffer runs out under different network interferences, especially packet loss and delay. As a consequence, we want to investigate how these parameters influence the perceived quality of video received by end users. An ISP may have partial influence on these characteristics and therefore may be able to tune the quality of video transfer and influence its users' satisfaction.

Our analysis revealed that there exists some small differences between the Flash and HTML5 strategies, however, in the most cases they will remain unnoticed by the end user. The buffering algorithm used in an HTML5 player showed the highest resilience against the packet delay and loss when taking into account the number of re-buffering events experienced during the video play. However, when comparing these both strategies with the Simple strategy, it is obvious that starting the video playback as soon as a minimum buffer level is achieved is insufficient. Although the Simple strategy has lower stalling time compared to the other two strategies, nonetheless, the number of re-buffering events which occur during the video streaming in unacceptable for an end user. All the three playing strategies cannot satisfactorily cope with even small limitation in the network throughput. The initial buffering mechanism implemented by the Flash and HTML5 strategies fails in that situation.

### 1.3 Organization of the Thesis:

In this thesis we are trying to measure youtube video packet loss when we playing video via internet with low bandwidth.

**Chapter 2**: Delivery Infrastructure, Cache Selection and Application-Layer
Traffic Management

**Chapter 3**:Video Distribution

**Chapter 4**:Previous work

**Chapter 5**:Experiment

**Chapetr 6**:Results

**Chapter 7**:Provides the conclusion with work and some ideas in further work in filed

# Chapter 2

## Delivery Infrastructure, Cache Selection and Application-Layer Traffic Management

Recent studies show that videos streaming sites represent about half of the Internet data volume, both for mobile access [3] and for fixed access [4]. Moreover, video streaming today produces a traffic volume that is more than double the one due to peer-to-peer. In the case of peer-to-peer the server resources to satisfy the demand come from the peer themselves, while in the case of video streaming a content distribution network must be built that consists of geo-distributed servers and caches. It is therefore instructive to study in detail the organization of a video streaming service and its evolution overtime. In the following, we will present the design principles of the YouTube delivery architecture and its performance.

### 2.1 Basic Facts about YouTube

YouTube, which was created in 2005, allows users to upload and share video content. Its success was immediate, resulting in spectacular growth ever since. For instance, the number of videos viewed per day has increased from around 200 Million in 2007 to more than 6 Billion in 2014 [5].


Fig 2.1:Watching Youtube March 2011-April 2012

Since YouTube was acquired in late 2007 by Google, its infrastructure has been in constant evolution and the delivery architecture that initially used third party content distribution network services is now fully operated and managed by Google. Not much about YouTube has been disclosed by Google itself [5–7]. However, in the last couple of years YouTube has been extensively investigated [8– 13] by academia via active and/or passive measurements, which are often carried out from multiple vantage points. While such measurements can reveal certain aspects of YouTube, many details are still unknown. Our description of YouTube is based on there results published in literature and on recent studies of YouTube carried out by ourselves.

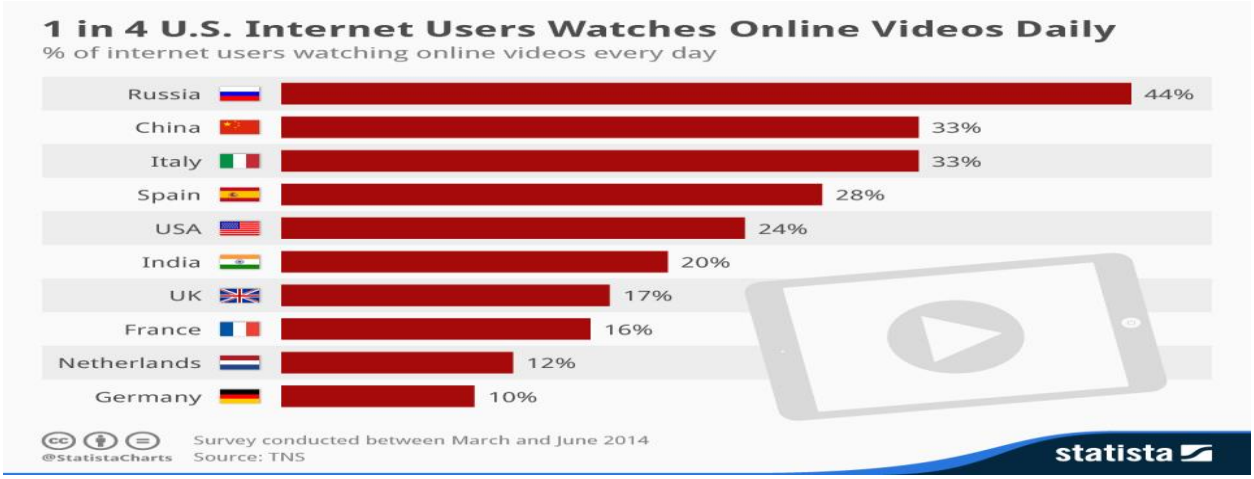



Fig 2.2:Statistics of online video user

Describing a system like YouTube that constantly evolves is challenging. However, we believe that the underlying design principles will most likely stay the same for some time. We sometimes give real figure to indicate the size of YouTube with the aim to give an idea of the order of magnitude and to provide a reference point for future comparison.

The number of videos hosted by YouTube was estimated in mid-2011 [14] to be about 500 million, which represents about 5 PetaBytes of data considering an average size of 10 Bytes per video. Taking into account replication of videos and multiple formats, this makes a total of about 50 Pita Bytes of data. In 2012, an average of *one* hour of video was uploaded every second, which is a three-fold increase as compared to 2009/ The number of videos downloaded per day has been evaluated in 2011 to be between 1.7 and 4.6 Billion representing a 50% increase over the previous year, which results in tens of Peat Bytes of traffic per day.

While YouTube originally started its service in the USA, it has become truly international in the meantime with caches in many countries. Today only 25% of the views are generated in the USA, followed by countries such as UK, Japan, Germany or Brazil, which each generate between 3–7% of all the views [5]. Geographic request locality is high, with around two thirds of the views per video coming from a single region, where a region represents a country or another political, geographic or linguistic entity [6]. Request locality also varies between countries and is highest in Japan and Brazil. In the following, we focus on the process of downloading the videos. We are going to review the major steps in accessing a YouTube video before we describe the server infrastructure and explain cache selection and cache redirections, which are used to balance the load among caches.

When a user uploads a video to YouTube, the video is stored on a server in one of Google backend data centers. YouTube supports multiple video formats. Each video may be transcended in all the different formats, which can happen pro-actively or on the fly. As we will see in the following, a user that requests a YouTube video will never directly interact with the servers in backend data centers. Instead, the videos will be delivered to the users from so called **caches**.

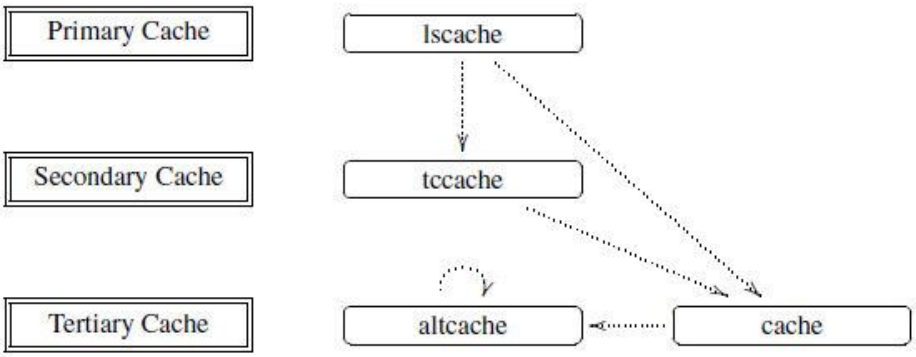



**Table 1:Cache classification in youtube**

## 2.2 Steps in YouTube Video Download:

Watching a video on YouTube involves a different set of servers. Initially, the embedding Web page is delivered through front end YouTube web servers, whereas the video content is itself

delivered by YouTube video cache servers. YouTube currently supports two containers for video streaming, Flash and HTML5 [14]. At the time of writing, the adoption of HTML5 for YouTube playback on PCs is still in an early phase, and almost all browsers use Flash technology as the default to play the YouTube videos [11]. When the container is Flash, a dedicated Shockwave Flash player must be downloaded to control the Flash plug-in in the browser. Simplified Steps in Accessing a YouTube Video**.** As shown in Figure 1, the process of accessing a YouTube video can be summarized as (numbers correspond to the graph):

(1)The user requests a video on the YouTube webpage:

https://www.youtube.com/watch?v=videoid and gets to the Web server that delivers the YouTube HTML page;

(2) After downloading the embedding HTML web page, the other contents are requested in particular the Shockwave Flash Player (embedded in a HTML object that contains the video parameters);

(3) The actual video content is requested from a cache server (Iscase server); if this cache is over-loaded, it sends a redirect (HTTP 302) message to the client indicating another cache server;

(4) The client sends a request the other cache server (tccache server) for the video,

and the FLV file is delivered to the client while being played in the Flash player

(Progressive Download). The details of the redirections depend on the load of the cache servers and are explained in the following .We now focus on the video content delivery, and more specifically on the architecture and the interaction with the cache server infrastructure.

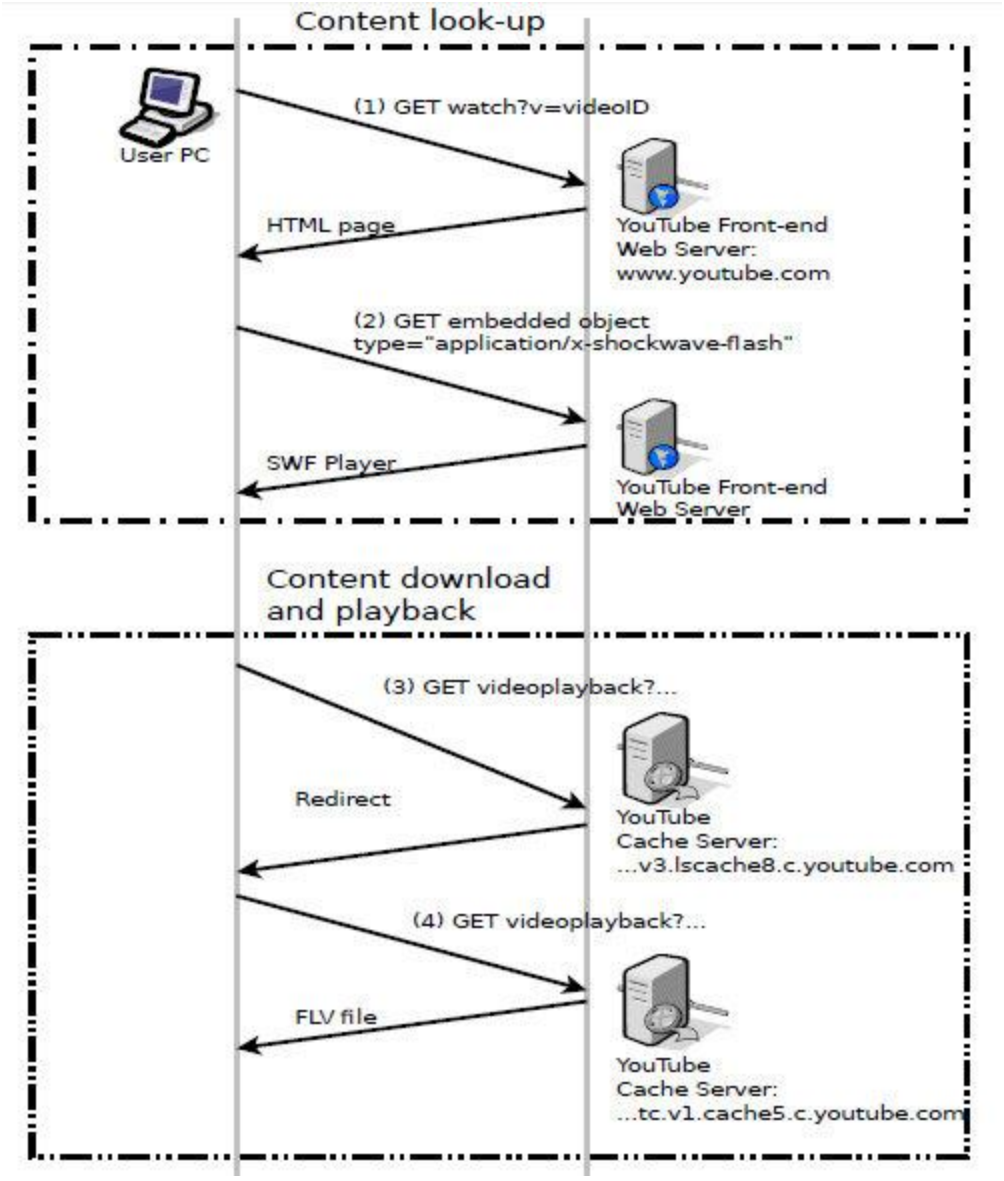



**Fig.2. 3.** Schema of YouTube page download

YouTube Cache Server Infrastructure**.** The users receive the video they request from a cache node. Individual **cache nodes** are organized in cache clusters, with all the machines of a cache cluster being co-located. The number of machines per cache cluster is highly variable and depends, among others, on the demand for service issued in the region where the cluster is located and also on the available physical space to host the cache nodes. Each cache node as of 2011 has a 10 Gb/sec network access and 78 Bite of disk storage [6].The various cache clusters are organized in a three tier hierarchy.

The global infrastructure of the YouTube caches has been revealed by Adhikari et al. [7] in 2014. They used the distributed infrastructure of the Planet Lab network to request thousands of videos from different vantage points in the world, which allowed to reverse engineer the cache infrastructure and the cache selection policies. We complement their findings with our own active measurements [16] undertaken in 2011 and 2012 from France. Our analysis focuses on residential Internet access and reveals the techniques applied by Google to deliver the videos to residential customers. Since our machines where connected to the Internet through different ISPs, we were able to observe differences in treatment of customers coming from different ISPs. YouTube has a three tier caching infrastructure that comprises of four different logical namespaces as shown in Figure 2. The machines allocated to the different cache clusters are identified via particular naming conventions. We recall here the main findings of [7] on the YouTube cache clusters. As of 2014 there are:

– 45 primary cache clusters with about 5000 unique IP addresses corresponding to the Is cache namespace;

– 8 secondary cache clusters corresponding to the cache namespaces with about 650 IP addresses;

– 5 tertiary caches clusters corresponding to the cache and alt cache namespaces with about 300 IP addresses. All these cache clusters are located in a total of 47 different locations distributed over four continents; there is no cache cluster in Africa. About ten primary cache clusters are co-located inside ISPs and the IP addresses of these cache nodes in these clusters are not part of the address space managed by Google but part of the ISPs address space. Since we have 38 + 8 + 5 = 51 cache clusters but only 47 different locations, some cache clusters belonging to different levels of the hierarchy must be at the same physical location (*i.e.* some primary and secondary caches are co-located). For the caches in each cache cluster, a particular logical naming structure is applied Each primary cache cluster has a total of 192 logical caches corresponding to the Iscache namespace, which looks as follows: city code v[1-24]. Iscache [1-8] .c.youtube.com As city code the three letter code for the airport closest to that cache cluster is used.

– There are also 192 logical caches in each secondary cache cluster, corresponding to the cache namespace, which are named as follows tc .v.[3-17] . cache [9] .c. youtube.com

– Each tertiary cache cluster has 64 logical caches corresponding to cache and attaché namespaces.272 T. Hoofed et al. Introducing these logical name spaces has the following advantages:

– Each video ID can be deterministically mapped via consistent hashing onto a unique logical name in the Is cache namespace, which makes it easy to decide for each cache what portion of the videos it is responsible to serve.

– There is a one-to-one mapping between the Is cache and its cache namespace.

– The logical naming is the same for each cache cluster and it is completely independent of the number of real cache nodes in a particular cache cluster.

– It is the responsibility of DNS to map logical cache names onto the IP addresses of real cache nodes. In [9], each of the logical names from the Iscache namespace is mapped to more than 75 different IP addresses distributed over the 38 primary cache clusters.

YouTube Datacenter Sizes. We have carried out active measurements in France [16], using simultaneously nine different Internet accesses (7 ADSL and 2 Fiber) to request videos during sessions that lasted for 2 days each. All these accesses are in the same physical location and the access rates for all the ADSL accesses are the same. The YouTube videos crawled during these measurements were served by two datacenters: one in Paris par, the other in Amsterdam ams. In Tab. 1 we show the number of IP addresses seen for each datacenter and match each IP address with its corresponding is a cache namespace. Tab. 1 gives an idea of the size of a cache cluster and also shows the evolution of these cache clusters over time:

– The Amsterdam cache cluster increased its size by 50% within a few months; for this cache cluster there are more IP addresses than distinct Is cache names, which means that a single Is a cache name will be mapped onto several IP address.

– The Paris cache cluster re-organized the distribution of its IP addresses into *two* distinct logical Is a cache namespaces. For this cache cluster there are fewer IP addresses than distinct Is a cache names, which means that several Is cache names will be mapped on the same IP address.

In Figure 2, we also show the dynamics of redirections inside the YouTube cache layers.

Each cache layer can redirect to the next cache level and the tertiary cache layer can be accessed through redirection out of any layer (including itself). We will explain this in detail in the next section on cache selection.

## 2.3 YouTube Cache Selection

YouTube cache selection is quite sophisticated and tries to:

– Satisfy users by selecting a nearby cache cluster and Perform internal redirection to another cache cluster to perform load balancing among cache clusters. The choice of a close-by cache cluster (in terms of RTT) is typically done through DNS resolution. DNS is used for coarse grained load balancing, with a TTL of five minutes. Before getting into the process of cache server selection through redirections, we first study the selection of first cache server.

Choice of First Cache Server With our active measurement carried out across different ISPs in France [16] we also wanted to investigate if clients from different ISPs get directed to the same cache cluster or not. We only focus on the first cache server returned and do not take into account redirections. All the accesses are located in the same place with the same access rate for ADSL.;

– ISP N has all its Is cache names pointing to the Paris cache site, but with two

Different logical name spaces (par08s01 and par08s05);

– ISP O has dedicated Is a cache names carrying the IPS name (Isp -par 1). Also,

these names get resolved to IP addresses that belong to a specific AS (36040),

This is different from the Google or YouTube ASes:

– ISPs S and F are directed to both cache clusters in Paris or Amsterdam with different proportions: about 2⁄3 to Amsterdam for ISP S and 10% for ISP F. These results highlight that there is a customization done for each ISP for reasons only known to Google. The network impact of the cache cluster location on the ping time is found to be very low. For example, the minimum ping time from our lab in France to the Paris cache nodes is of 23.8ms and of 28 ms to Amsterdam (because of relatively small distance between the two cities). However, the main point is that the cache cluster selected in not necessarily the geographically closest one and that the choice of the preferred cache cluster depends on the time of day as shown in Figure 3 and on the ISP the client is connected.

Moreover, even if the minimum ping values for both cache clusters are about the same, the cross traffic on the path from France to Amsterdam can increase the ping value to values as high as

200ms. Indeed, Figure 3 shows a large variance in ping times towards Amsterdam cache nodes. The most striking point is that the switch from one datacenter to another is done at a specific time every day, and this time is specific to each ISP. We have made the same observation in [16] for a different experiment with clients located in Kansas, USA, who were served from a variety of different cache clusters located anywhere in the US. In Figure 4, we present a partial map of USA with the location of the most prevalent cache clusters seen in this crawl. The symbols have the following meaning:

– The pin mark is the place from where the crawls are performed: Kansas City;

– Each circle corresponds to a YouTube cache site;

– The diameter of each circle represents the number of videos served by this cache site;

– The color of each circle represents the distance (ping time) towards the cache site: green for ping time lower than 60 ms, blue for ping time between 60 and 200 ms, and red for ping time larger than 200ms.

Note that we do not show the San Jose and Los Angeles cache sites in the figure, which receive 11% and 3% respectively of the video requests. There are four more cache sites, which are located in Kansas-City, Lawrence, Chicago and New-York that receive a small fraction of all requests. The details can be found in [15]. We clearly see that the distance is not the primary criterion for the choice of the cache site: the most frequently used cache site is in Washington DC, even though it is much further away than the of the circle, and distance (circle color: green for ping $\leq$ 60 ms, blue for ping $\geq$ 60 msand $\leq$ 200 ms, and red for ping $\geq$ 200 ms of the YouTube cache sites in the crawls that originate from Kansas City (pin mark).

### 2.3.1 Cache Selection through Redirections:

We have already seen that load balancing on YouTube cache servers can be done through DNS resolution: this process is centralized at the YouTube authoritative DNS servers. Load-balancing can also be done directly at cache server level. In this case, the cache server receiving the request can relay the request to another cache server via HTTP redirect message at application level. So YouTube can use both centralized and decentralized processes to balance the requests on its cache servers. Cache hit. If the video is hot and there are copies at the primary caches, then a

logical cache node (Isa cache namespace) in the primary cache is chosen. If there is no redirection, a machine from a cache cluster serves the video. If the primary cache cluster selected is overloaded, a redirection to a secondary cache cluster (cache namespace) occurs. The secondary cache can serve the video or redirect it to a tertiary cache site (cache namespace) for load-balancing purposes. Again, in the tertiary cache cluster, the cache server can deliver the video, or perform another redirection.

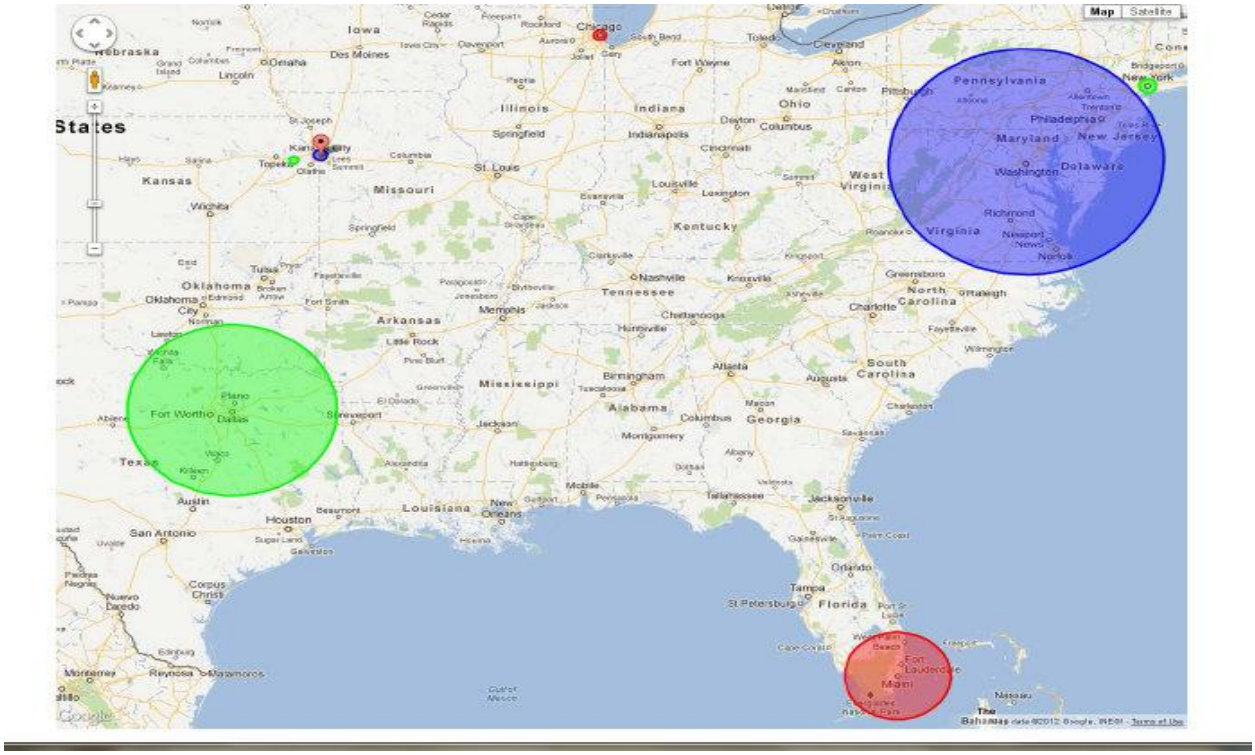



Fig 2.4:Map indicates the locations cache site, number of request served by each ammeter of the circle and distance (circle color: green for ping ≤ 60 ms, blue for ping ≥ 60 ms and ≤ 200 ms, and red for ping ≥ 200 ms) of the YouTube cache sites in the crawls that originate from Kansas City (pin mark).

A redirection from a tertiary cache site will remain inside the tertiary cache level and ocur towards a cluster from the altcache namespace. A machine in this namespace now serves the video or redirects it inside the same namespace in case of overload. Very rarely, several redirections occur inside the altcache namespace, with the total number of redirections being limited to 9. For further detailssee [8].

Cache Miss. If the video is cold and there are no copies at the primary caches, then the request will be most likely redirected from the first level cache to a third level cache. The third level cache will fetch the video from the backend data server, cache it, and deliver the video to the client. It is quite common, as we will see in the next section that users do not watch the entire video. Therefore, all videos are broken into chunks(of 2MBytes) and the cache will continue to retrieve from the backend servers new chunks of a video as long as the user keeps viewing that video. Note that despite all the efforts of the engineering team of Google, the cache miss rate remains steadily at about10% [5].

### 2.4 YouTube Redirection Process:

DNS Level Redirections. YouTube uses HTTP redirections to balance the load among its caches. As shown in Figure 2, the redirections usually direct the video request from a cache layer to the next one. Using traces from a European ISP, Torres et al. [13] observed that as the total number of requests kept increasing, the percentage of requests handled by the closest cache cluster located inside that ISP decreased from 80% to about 30%. In this case, DNS request resolution will direct clients to more distant but less loaded cache clusters. Impact of Redirections on Performance. Each redirection involves:

1. DNS query to resolve the hostname of the next cache node,
2. Opening of a new TCP connection,
3. Issuing a new video query.

In case of redirections, the final cache node serving the video will most likely not be the closest in terms of RTT, which has been observed in [14] for the most popular videos of the day. The impact of redirection on the time until the first byte is downloaded (referred to as video initialization time) has also been studied in [8]. The video initialization time is on average 33% higher if the video has been fetched through redirections. The fraction of sessions that have been redirected is evaluated in [11]: between 10% and 30% of all sessions are redirected at least once. The impact of redirections on the startup delay can also be important [11]:

– Without redirections, delays are in the order of milliseconds;

– With redirections, delay can increase by orders of magnitude, up to 10 seconds.

## 2.5 Application-Layer Traffic Management

YouTube videos are requested using HTTP over TCP. TCP is a transport protocol that assures reliable transfer by retransmitting lost data packets and performs congestion control to avoid overloading the network. Both error control and congestion control of TCP may result in high delay jitter. The delivery strategy of YouTube videos has been studied in great detail by Rao *et al.* [15]. The authors show that the delivery strategy depends on the video container (Flash, Flash High Definition, or HTLM5), the type of client device (PC or mobile devices such as smart phones or I Pad), and the type of browser (Internet Explorer, Chrome, or Firefox). The delivery strategy needs to reconcile a number of potentially conflicting goals such as:

– Smooth play out during the entire duration of a viewing session;

– Efficient use of the server resources such as disk I/O and timer management;

- Avoid to transmit too much data in advance of consumption in order to (i) reduce the amount of buffering at the client, which is particularly relevant in the case of mobile devices and to (ii) reduce the waste of network and server resources by sending data that are never used. Find more *et al*. [11] observed that 60% of the videos requested were watched for less than 20% of their total duration, resulting in an un-necessary transfer of 25–39% of the data. As we shall see in the following section, the impact of playback degradation is a primary factor in the video transfer interruption. As the video transfer is done via HTTP over TCP, there is not guarantee that the data can be delivered to the client at the rate at least as high as the one at which they are consumed. The details of the transfer have been studied in [15], whose findings we summarize in the following: To increase the likelihood of a smooth playback, YouTube performs aggressive buffering when a video is requested. Initially, during a startup phase, the server sends as fast as possible to fill up the initial client play out buffer. This play out buffer contains about 40 seconds with Flash, and 10–15MBytes with HTML5 with Internet Explorer as a browser, which is typically much more than 40 seconds worth of video. Once the initial buffer has been filled, two other strategies are used by the cache server:

– keeps sending as fast as possible, until entire video is transmitted;

– limits the rate of the transfer alternating between on-off cycles with a fixed period.

During an on cycle, a fixed size block of video data is transmitted. We limit our description to the case of streaming a video to a PC with Flash as container, and refer to the original paper [15] for more details. Streaming the video to a PC has been the most extensively studied [7, 17]. In this case, the server streaming strategy is independent of the browser: When the startup phase is terminated, the cache server sends blocks of 64KBytes at a frequency that allows to achieve an average transmission rate of 1.25 times the video encoding rate. As has been first observed by A lock and Nelson [17], injecting bursts of 64KBytes means sending 45 maximum size TCP segments back-to-back into the network. Such large packet bursts will accumulate in the buffer of the bottleneck link and (i) cause Internet Video Delivery and QoE 279 delay spikes that may disrupt other latency sensitive application and (ii) inflict loss on the burst YouTube flow itself. In response to these problems, Google engineers have recently proposed a modification to the server side sending policy that controls the amount of packets that can be injected back-to-back in order to limit the size of the packet bursts. For details of the new sender algorithm and its impact on packet loss and burst size see [7].

In this section we have provided the technical basis to understand YouTube content delivery over the Internet. Next, we investigate what influences the QoE experienced by the user. In particular, problems in the network may lead to stalling and QoE degradations. Therefore, we have to identify the key factors that influence YouTube QoE by means of subjective measurements and build an appropriate model, which can be used for QoE monitoring later on.

# Chapter 3

# Video Distribution

## 3.1 Protocols

Contemporary media delivery systems can be classified into two categories: systems with and systems without feedback control mechanism. One of the options for multimedia delivery systems with

Feedback control is an usage of the Real-Time Streaming Protocol (RTSP). The RTSP is a state full protocol, which means that the server keeps track of the client's state from the first time the client connects to the streaming server until the time it disconnects. The client communicates its state to the server by sending commands such as play, pause or disconnect. The server begins sending the media as a steady stream of small RTP (Real-time Transport Protocol) packets. The data is sent at the media bit rate and the client buffer is filled with just few packets before playback begins. If the client or servers discover any interference in their communication, like increasing latency or packet drops, they can renegotiate transmission parameters, e.g., the server can send the same video content but with reduced encoding rate. The transmission is usually based on unreliable transport protocols, most commonly the UDP. However, when using the UDP, data packets often have difficulty getting around firewalls and network address translators. Thus, sometimes the TCP is preferred when firewalls or proxies block UDP packets, although at the expense of potentially unnecessary reliability. Such problems are limited when using the HTTP as a media delivery protocol because firewalls and routers know how to pass HTTP traffic through. It also does not require special proxies or caches. The HTTP is a stateless protocol. Thus, multimedia transmissions based on it share this feature and behave as a system without feedback control. Basically, if an HTTP client requests data, the server responds by sending the required data, but it does not remember the client or its state which means that each HTTP request is handled completely independently. HTTP streaming may be implemented in several ways. In our work we focus on an implementation which can be described as a progressive download. The progressive download is nothing more than a transfer of a video file from a

HTTP server to a client where the client may begin playback of the file before the download is complete. Contrary to the above mentioned systems with feedback control, which rarely send more than a few seconds of video content to a client in advance, HTTP streaming (web) servers progressively push the whole video content to a client, usually does not taking into account how much data have been already sent in advance. Simultaneously, most players are capable of playing the video file while its download is still in progress. Most web-based streaming platforms, including Video, MySpace, and MSN Soapbox, are based on HTTP and do not have a feedback control. However, some HTTP streaming services, e.g. YouTube, implement additional application layer flow control mechanisms that limit the transmission rate to the same magnitude as the video bit rate [18].Currently, it is thought that HTTP media streaming is easier and cheaper to deploy because web streaming can use generic HTTP solutions and does not require specialized servers at each network node. Standard HTTP caching mechanisms allow to move media content to an edge of the network, closer to users. Nonetheless, the above technologies have also shortcomings. The congestion avoidance algorithm of the TCP produces a saw-tooth shaped transmission rate. Furthermore, the reliability of TCP results in variable transmission delays due to retransmissions of lost packets. As a consequence, it was commonly assumed that the TCP is not suitable for multimedia streaming, which is to some extent loss tolerant but delay sensitive. The instantaneous transmission rate and transmission delay variation of the TCP must be smoothed out by receiver-side buffering. Despite these drawbacks, currently a dominant share of multimedia traffic is being delivered using the HTTP and TCP [1].

### 3.2. Video Buffering

Most of HTTP players are able to concurrently play and download the same file. In the simplest case the player fills its internal buffer at the beginning of the video transmission and starts the video playback as soon as a minimum buffer level is achieved. While simultaneously playing and downloading the content, the amount of video data in the buffer is variable and depends mainly on the download bandwidth, video bit rate and video playing rate. When the download bandwidth is larger than the video rate the buffer grows. In the opposite case, the buffer will shrink and if the situation last long enough it may also run out. In such cases the video stalls and the player waits until the buffer will be refilled again.

Let's assume that $G(t)$ represents the number of data packets generated at a HTTP server by time $t$ , Fig. 1 with a packet transmission rate limited only by the infrastructure conditions like TCP throughput, server performance, etc. The first packet is generated at time 0 and sent immediately to a client. Let $A(t)$ denote the number of packets arriving at the client by time $t$ and $B(t)$ denote the number of packets played by the client by time $t$ .

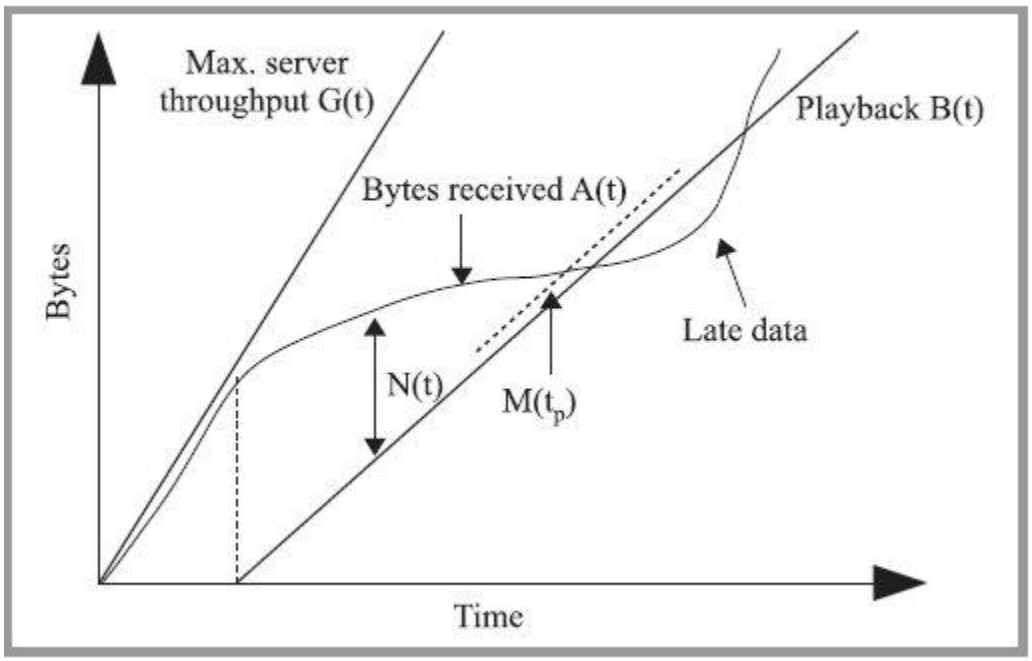



**Fig 3.1.** Player buffer occupancy in the time function.

Since the transmission rate is constrained by the generation rate at the server, we have $A(t) \leq G(t)$. A packet arriving earlier than its playback time is referred to as an early packet. At time $t$ , the number of early packets is counted as $N(t) = A(t) - B(t)$.A negative value of $N(t)$ indicates that the packet arrival is behind the playback by $-N(t)$ packets. During streaming, there can be many time periods $\Delta ti$ for which $N(\Delta ti)$ has a negative value. In our work we try to answer the question: what is the value of $\sum i \, \Delta ti$ and $i$, i.e. the total video stall time in a relation to video clip length and the number of stalling events for several video files transmitted from YouTube.

### 3.3. Video Playing Strategies

For video streaming YouTube currently uses amongst others device-dependent 3GP containers with RTSP dedicated for mobile streaming applications and Adobe Flash with HTML5 employing HTTP streaming of Flash Video. Adobe Flash, henceforth referred to as Flash, is the default container when YouTube is accessed via a PC. Users need to install a proprietary plug-in for viewing Flash videos.HTML5 supports videos that do not require any proprietary plug-ins running directly.

When streaming with the Flash Player, it basically behaves like a simple HTTP player described above i.e. it starts the video playback as soon as a minimum buffer level is achieved. However, thanks to the flexibility of the Flash authoring platform, the buffering functionality can be additionally enhanced using client-side Action Script code. The standard buffering process is believed to be susceptible to bandwidth drops, as well as being unable to exploit a sudden increase of bandwidth. The enhancement is called a dual-threshold buffering strategy and assures a faster start and, at the same time, should provide better resilience to bandwidth fluctuations, or other adverse network conditions. Therefore, the playback of a video file starts when the first threshold in the buffer is filled with a given amount of data. But, instead of trying to keep the buffer full to this level, the modified strategy attempts to fill the buffer to a second, higher threshold. This additional data may be useful later if the network connection encounters temporary impairments like bandwidth drops or fluctuations. In the case of HTML5 streaming, the playing strategy depends on particular video player implementation. TheW3C HTML5 specification [4, Section 4.8] states, that in the case of auto play "the user agent will automatically begin playback of the media resource as soon as it can do so without stopping". To approximate this difficult to full fill condition every implementation differs. We investigated Firefox's implementation of this spec as its code is open source and the behavior can therefore be studied not just by observing network traces but also by reading the sources.

The algorithm in the Firefox is summarized in algorithm in Fig. 2 and Table 1. Rather than using static thresholds it facilitates moving averages to estimate the development of the transmission rate. It does not differentiate between the initial video startup time and intermittent buffering events.

If $S_{MA} > V_{MA}$ **then**

$c \leftarrow (b_b = 20s \vee b_r = 20s)$

**else**

$c \leftarrow (b_b = 30s \vee b_r = 30s)$

**end if**

This implementation requires large playback buffers due to the chosen high video buffering amounts, but could also result in very few stalling events.

| Variable | Explanation |
|---|---|
| $S_{MA}$ | Moving average of the transmission speed. |
| $V_{MA}$ | Moving average of the video bit rate. |
| c | Condition upon which to start/resume playback. |
| $b_b$ | Amount of video data the buffer contains. |
| $b_r$ | Amount of time spent in non-playing buffering state. |

Table 2

The HTML5 network traffic also differs from the Flash traffic. The works [5] and [2] identified YouTube's block transmission behavior, which uses longer and client application controlled block phases for Google Chrome and no blocking at all for Firefox.

# Chapter 4

## . Previous Works

A major research area related to our work is concerned with the analysis and characterization of streaming services in the Internet. Early works in this area go back to the twentieth century and focused amongst others on the characterization of videos on the Web [21], video access statistics of users [22], developing UDP-based streaming protocols and providing mechanisms for TCP-friendliness and loss recovery, e.g. [23], [24]. When to concentrate on the HTTP video, several YouTube measurement studies have been reported in literature in the last few years. These works focused on characterizing various aspects of YouTube videos, as well as its usage patterns. On the one hand, we have work based on user traffic trace analysis including deep packet inspection, e.g. [2], [26]–[27]. Their authors operated on real world measurements obtained from, e.g., ISPs' networks and they characterized video popularity, durations, size and playback bit rate, as well as usage pattern statistics such as day versus night patterns or traffic volume. Additionally, in [25] the investigation of YouTube user sessions statistics was presented. In [2] the authors presented a traffic characterization of Netflix and YouTube, and identified different streaming strategies deriving also a model for the aggregate traffic generated by these services. Plissonneau *et al.* in [27] described the impact of YouTube traffic on a French regional ADSL point of presence revealing that YouTube video transfers are faster and larger than other large Web transfers. On the other hand, there are publications based on crawling the YouTube site for an extended period of time [28]–[30]. These works examined video popularity and user behavior and found that statistics such as length, access patterns, growth trend, and active life span were quite different compared to traditional video streaming applications. Furthermore, in [28] information directly available from YouTube servers was used to analyses the characteristics of videos served by YouTube while [29] investigated social networking in YouTube videos. Also Abhari and Soraya in [30] investigated YouTube popularity distribution and access patterns through the analysis of a vast amount of data collected by crawling the YouTube API. On the basis of the observations, the authors presented essential elements of the workload generator that can be used for benchmarking caching mechanisms. A global study of user experience for

YouTube videos using Planet Lab nodes from all over the world is performed in [31]. Results from this analysis show that on average there are about 2.5 pauses per video, and on average 25% of the videos with pauses have total pause time greater than 15 seconds. The closest work to ours is [32] where the authors evaluated the responsiveness of adaptive HTTP algorithms (taking into account YouTube amongst others) under variable network conditions. The authors claimed that the performance of the streaming algorithm increases with the decrease of network delay and by providing information to the client, particularly about the achievable throughput. It compensates for the structural noisiness of measurements and improves the ability of the client to accurately estimate the throughput.

# Chapter 5

## Experiments

In order to observe the  of YouTube, and, for that matter, any other, streaming under varying network conditions, we work with cmd to get request a client computer simulate a streaming application by issuing HTTP GET requests to videos at a YouTube cache. Any video data is stored and analyses for its frame characteristics, such as the size, type and relative playback time. Further more ,the client captures the packet trace using capsa, allowing offline analysis of the packets to and from the HTTP server. The process of transmission monitoring by command promo, such as packet delay distribution, packet loss rate, or transmission throughput. Random packet losses are limited due to access network link layer and transport protocol retransmissions. We work with 356kbps broadband bandwidth.

Step 1:For checking the packet loss We just open 1 videos in our web browser and open CMD and type this command TRACERT WWW.YOUTUBE.COM" and hit Enter. Then we will see the delay and transmission of cache  process for this  video.

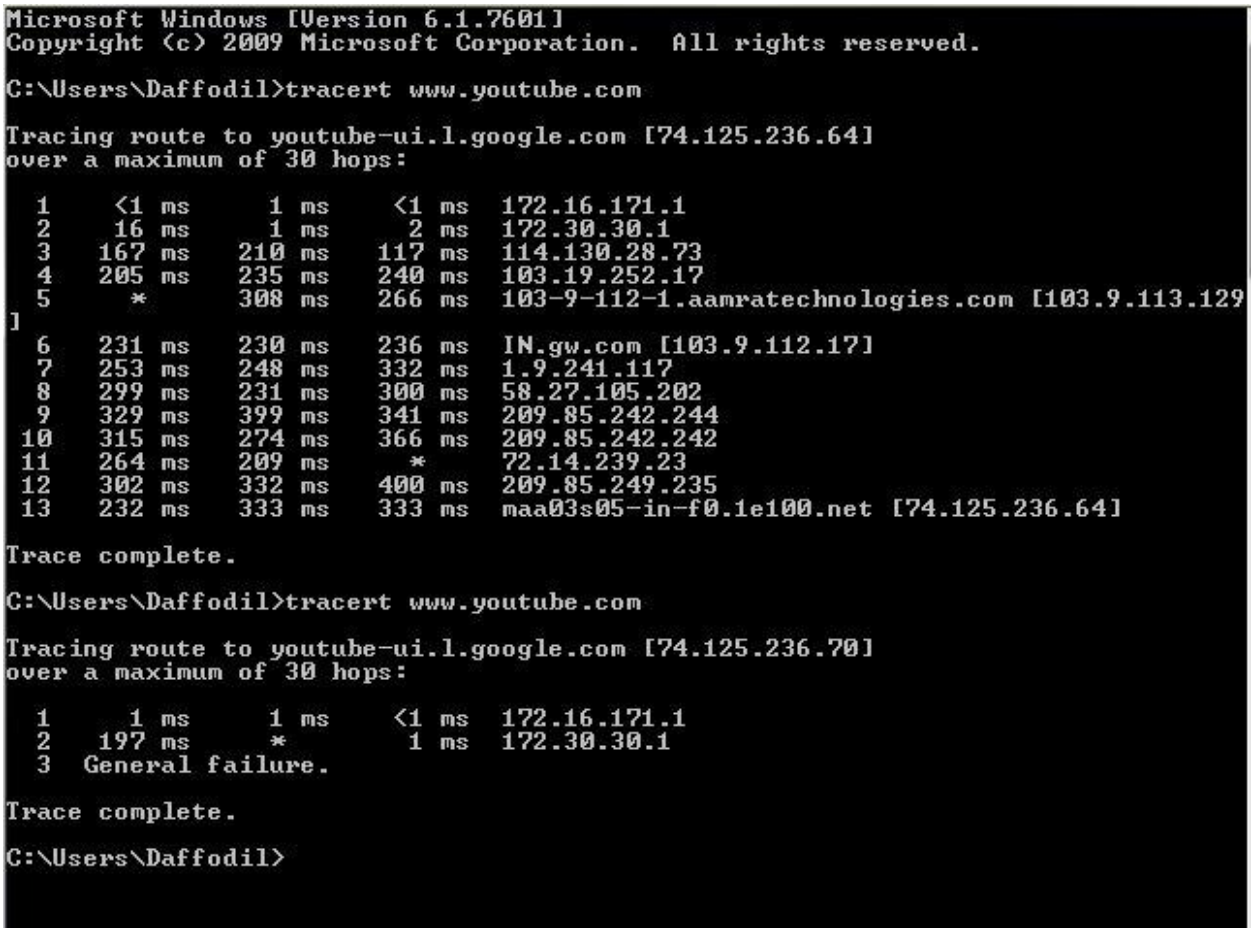



Fig 5.1:Tracert www.youtube.com by one video played

Now we try to check the packet loss and when its happened and how much. So first we need to ping the whole IP whose we get our previous command.

We ping three IP. First IP ,6th IP and 10$^{th}$ IP.

For ping just open the command prompt and type PING and SPACE First IP (172.16.171.1)

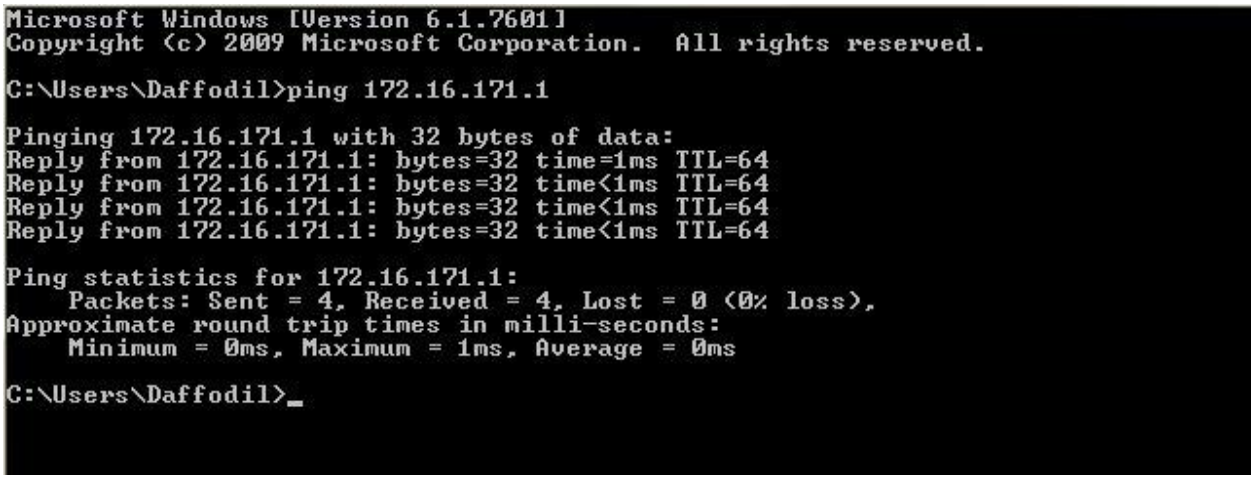



Fig 5.2:ping IP

We can see here is 0% packet loss because of first time video playing is to be continued without buffering but after 205 ms we got packet loss. Now we ping 6$^{th}$ IP in same way. Just type PING and hit ENTER.

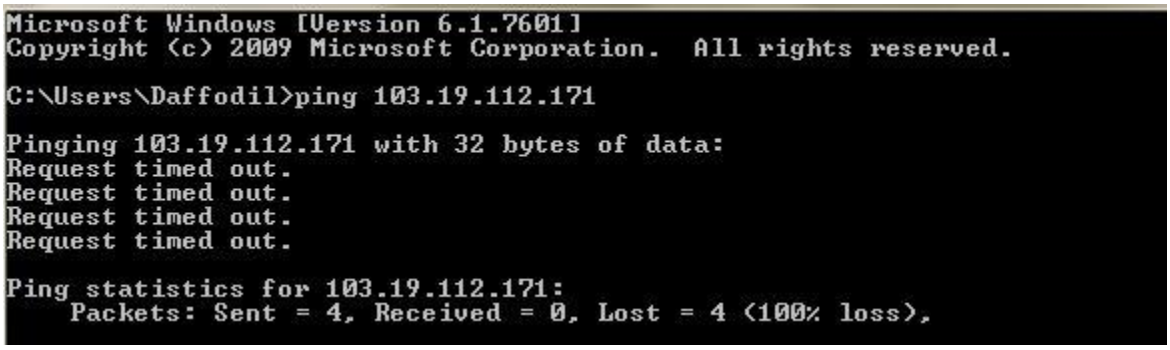



Fig 5.3:ping IP

Now we can see we get packet loss=4 here. Because after 230ms the video played by sending huge of packet and its showing packet loss.

Again we trying to pin 10th IP in same way.

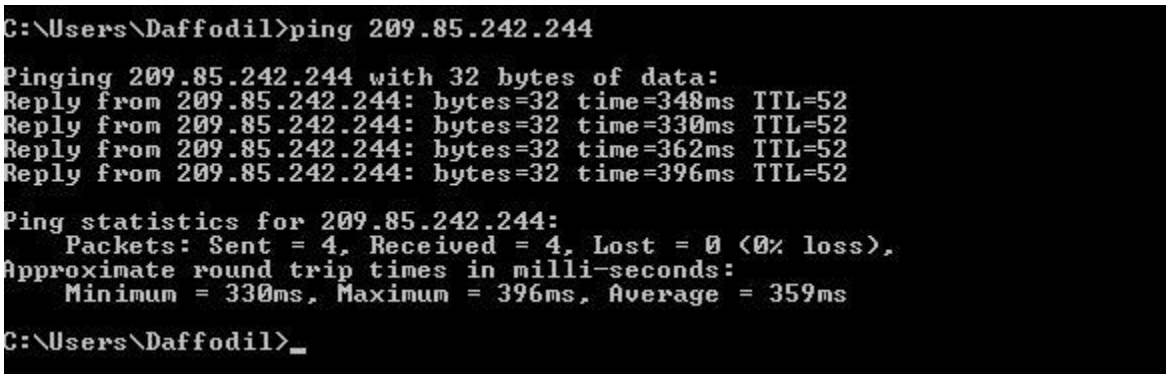



Fig 5.4:ping IP

Here is again no packet loss because video playing smoothly and transferring low quality video graphics.

Step 2:In step 2 we open about 4 tab of youtube in browser. Cause its huge bandwidth but we have only 356kbps.


Fig 5.5:Tracert again www.youtube.com

After TRACERT we get about maximum 30 hops tracing route and its increasing rather than before tracing. Now we take second IP and we get packet loss here after pinging. Because of for huge number of playing video ,packet loss start from first step.

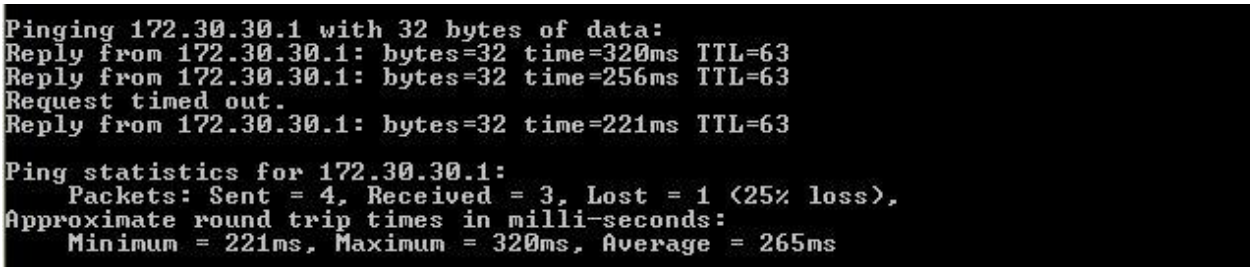



Fig 5.6:Ping IP

again form taking 4th IP we don't get any packet loss because its overcome the transferring huge of data error or transmission of packet was slow or video quality is not good.

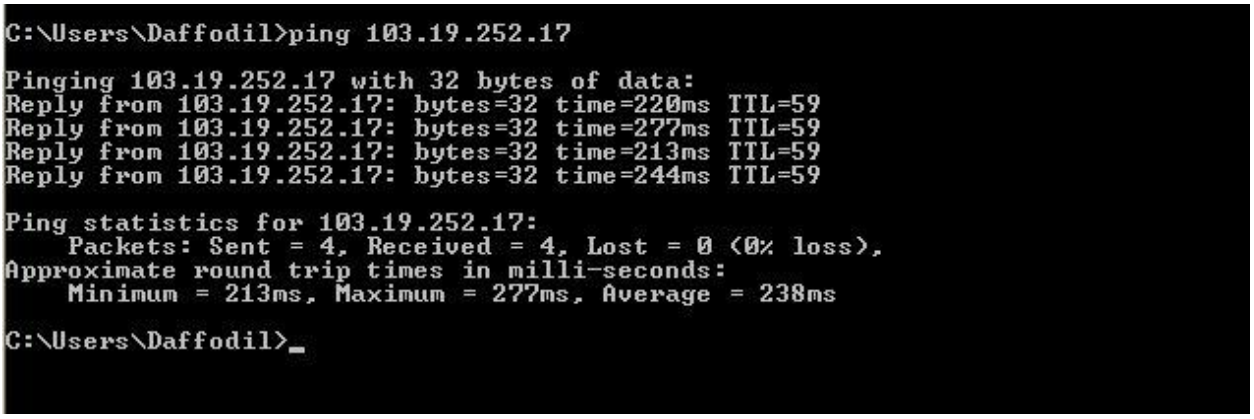



Fig 5.7:Ping IP

However, through these mechanisms, loss acts as another source of traces independently of playback strategies. Delay and jitter. We focus on asymmetric access networks, such as ADSL phone lines, which are assumed to form the bottleneck links. In the second phase, the video file with the frame dataset is re-assembled. Then they are used to feed models in a media playback emulation process, which combines the transmission and video frame traces to calculate the playback buffer fill level for every point in time during the playback. Each frame has its own playing time which specifies the time at which the frame should be played in relation to the initial frame. From this analysis, we can determine how much data is required to play each frame of the video without any delay to allow for an uninterrupted playback. On this basis, we generate statistics about user-perceivable artifacts such as re-buffering events that would occur during the playback. These statistics can then be compared to the results of other models and network QoS. For our experiment we used a 92 s video file encoded at 850 Kbit/ps.

### 5.1. Quality Measures

The characteristic of a network is the QoS of received multimedia content. However, in the case of HTTP video the transmission is reliable, so there is no packet loss induced video degradation. Nevertheless, packet losses introduce additional delay caused by TCP retransmissions which consequently can lead to re-buffering events resulting in jerky playback. The packet delay and loss reduce also TCP throughput. When the throughput is lower than the playback rate and the buffer has drained, the video playback will pause and wait for new video data. A user expects that delays resulting from content buffering will be minimized and do not occur during normal video play. Thus, to characterize the relationship between the network QoS and application QoS, for our purpose, we use two form the user's perspective, the key performance measures for HTTP videos. The first measure of the application QoS takes into account relative total stalling time experienced by a user and is defined as:

$$SR = \sum i \Delta t_i / T, \quad (1)$$

Where $t_i$ are times for which $N(\Delta t_i)$ has negative value and $T$ denotes a total duration of the video file when played without interruptions. As the above measure is the ratio of total stalling time to the video duration, it is desirable to minimize its value by an ISP. The application QoS defined in (1) did not differentiate between the cases in which a user can experience one long stalling period $\Delta t\ l$ or several shorter stalling periods $\Delta t^s$ where $\Delta t^l = \Delta_i \Delta t_i^s$ Thus, in our analysis we also use a second, complementary measure which value is the number of re-buffering events $i$ associated with every stalling period. In our experiment every video playing scenario has at least one re-buffering events which is a result of an initial buffering. The initial buffering is used to accommodate initial throughput variability or inter-packet jitters. Some streaming strategies may achieve smoother streaming with larger initial buffering, nonetheless it increases the startup latency of received video content. The re-buffering, which take place in the middle of video play back, are usually a consequence of the congestion avoidance algorithm of the TCP. The last strategy assumes that the algorithm always starts playback as soon as any data is available in the buffer. This means that, if the player is currently stalling and a complete frame becomes available in the buffer, playback will immediately restart and the frame will be shown even if this means stopping playback after that frame again. This results in the lowest required buffer space. Moreover, playing the video as soon as possible, gives the fastest end.

# Chapter 6

## Simulation and Results

### 6.1. Delays

The delay in an experienced by video content consists of two components: delay introduced by network, which is the time it takes a data packet to travel from sender to receiver and TCP-level delay, which is a consequence how the TCP reacts to fluctuations in the effective network throughput.

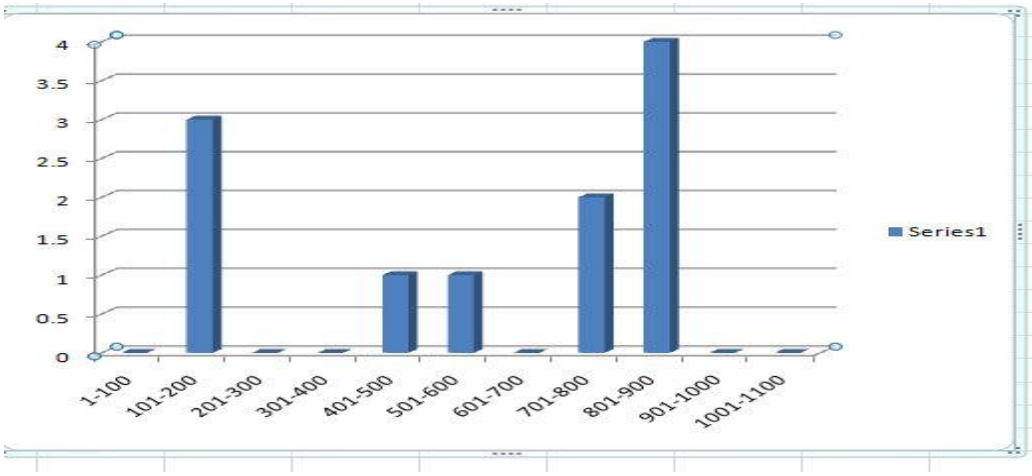



Fig 6.1:Histogram of first step of tracent

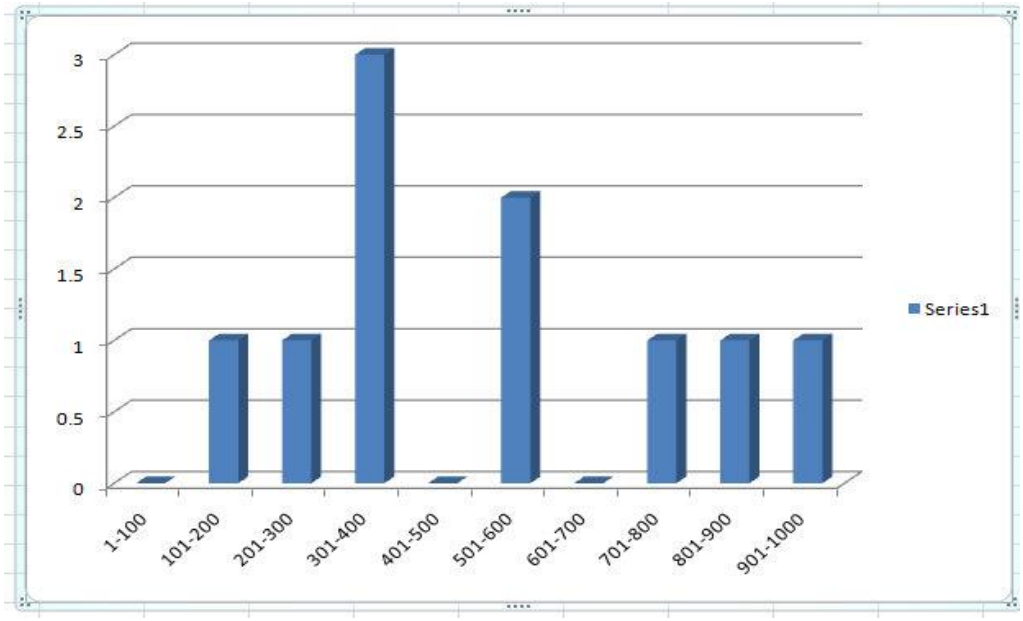



Fig 6.2:Histogram of second step of Tracert

While throughput fluctuations can occur due to application level flow control, they are primarily the result of network congestion.

As results of our experiments we obtained statistics of buffer occupancy as a function of time for the three examined playing strategies. An exemplary trace of the buffer occupancy for the Simple strategy is presented in Fig. 10.

We may notice that with increasing latency the buffer occupancy, measured as video playback time, is decreasing and re-buffering events happen more often. When the packet delay is 50 ms, there is only one stalling event on the beginning of the video transmission. According to the formula throughput ~ 1/delay describing theoretical TCP throughput, in this case the delay is the lowest, thus the theoretical connection throughput is the highest. Therefore, the download of the whole video finishes after about 80 s of the experiment. From this moment the buffer is no longer supplied. It decreases at a constant rate as the video player pulls the remaining video data and presents it to its user. However, when the delay rose to 100 ms or 200 ms, watching the video is quite inconvenient due to the frequent buffer under-runs. In these situations more re-buffering events occur, the total playing time exceeds the original video length and the file downloading completes after about 92 s. Generally, we were interested not in a transient buffer analysis but in its examination in the context of application QoS for which measures were

defined in the Subsection 4.1.Thus, in the further experiments, except for the packet delay, we obtained statistics of the buffering behavior in scenarios with additional packet loss and network throughput limitation. As it is shown in Fig. 5(a), packet delay has a certain influence on application QoS which is defined as the SR in Eq. (1). Increasing gradually the packet delay up to 1000 ms caused the successive rise of the SR from less than 1% to about 100% in average. From the three examined playback strategies, the Simple strategy experienced the lowest SR while the highest SR was obtained by Firefox HTML5 strategy. Nonetheless, with increasing delay, the differences between the playing algorithms diminished. When it comes to measuring the number of stalls, the situation looks quite different. When increasing latency up to 300 ms, a user using the Flash or HTML5 strategies usually experienced only a single re-buffering event which occurred at the beginning of the playback. When the delay exceeds 500 ms the HTML5 strategy has the lowest number of stalls from all the three examined strategies. The Simple strategy was not able to successfully mitigate the network impairments which resulted in several re-buffering events during the playback, Fig. 5(b).

### 6.2. Packet Loss

Packet loss can be caused by a number of factors including signal degradation over the network medium due to multi-path fading, packet drop because of channel congestion, corrupted packets rejected in-transit, faulty networking hardware, faulty network drivers or normal routing routines. When transmitting HTTP video, in the event of packet loss, the receiver asks for retransmission or the sender automatically resends any segments that have not been acknowledged. Nevertheless, retransmitting missing packets causes the throughput of the connection to decrease due to the sliding window mechanism used for acknowledgement of received packets, implemented in the TCP. For the packet loss up to 2% the SR graph resembles *S* shape, Fig. 6(a). For packet loss below 0.8% the SR has value 1 for the Flash and HTML5 strategies, and about 0.2 for the Simple strategy. Such values of the SR can be considered relatively low and should not have much impact on the received video quality. However, for the packet loss between 1% and 1.2% the SR rises rapidly achieving values of several tens. The further increase of the packet loss rate results in a relatively small rise of the SR. Generally, the

Simple strategy has the lowest value of the SR. We can also notice that for 1% packet loss there is a significant difference between the Flash and HTML5 strategies which diminishes for the other packet loss values. When to measure the video streaming resilience against the packet loss as number of stalls, Fig. 6(b), the shape of the chart is similar to the shape of the SR presented in the Fig. 6(a). We can also observe here the steady rise of the stalls number when the packet loss is lower than 0.8% and higher than 1.2%. For the packet loss value between 0.8% and 1.2% the stalls number grow quite fast. Contrary to the results presented in the Fig. 6(b), this time the Simple strategy has the worst performance. The HTML5 strategy is little better than the Flash strategy for the packet loss higher than 1%. In remove the power loss effect of image we can also use mat lab coding program.

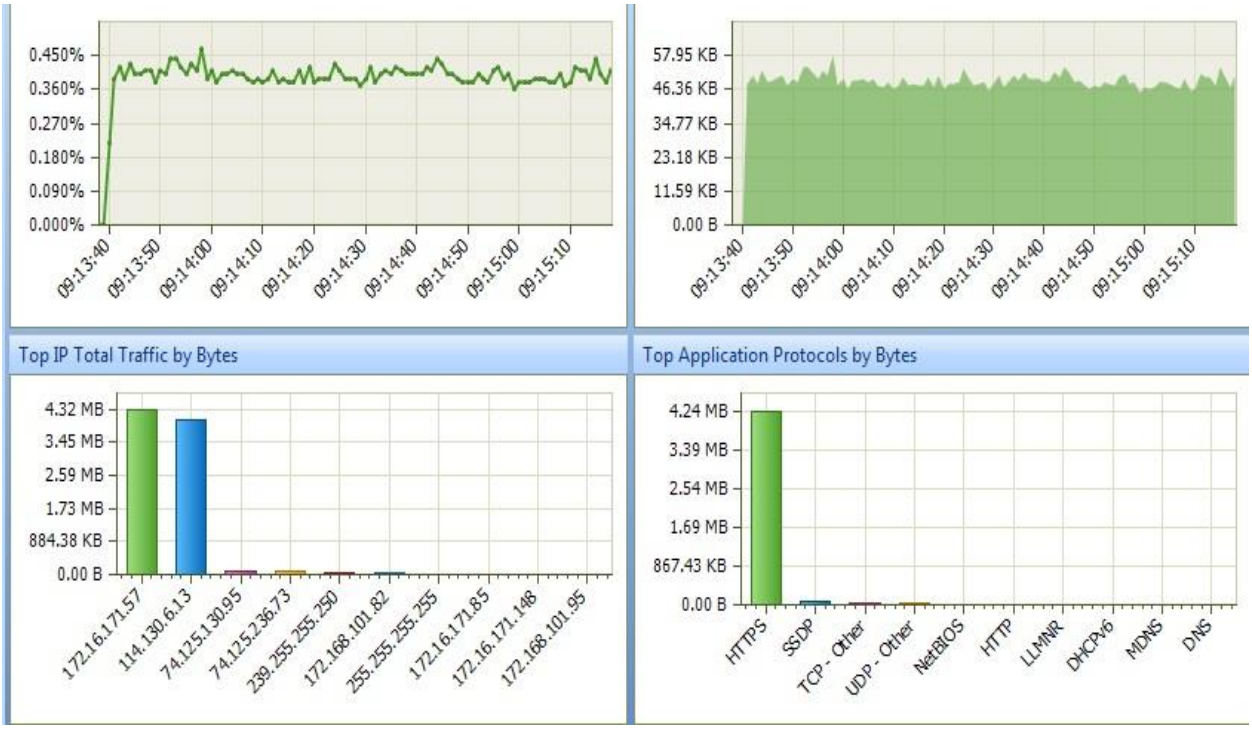
```
1.clear all

2.close all

3.cle

4.warning off

5.tic

6.frame=imread('lena.jpg');

7.imgfr=rgb2gray(frame);

8.rsize=128;

9.csize=128;

10.img=imresize(imgfr,[rsize csize]);

11.figure

12.subplot(3,3,2),imshow(img);

13.title('original image');

14.img=double(img);
```

```
15.bit_size=1;

16.blk_size=4;

17.img_in=img;

18.outimg=image_down_sample_process(bit_size,blk_size,img_in);

19.recon_img=image_recover_process(bit_size,blk_size,outimg);

20.error_val=mean2((img-recon_img).^2);

21.psnr=20*log10((255^2)/error_val);

22.subplot(3,3,2),imshow(uint8.(recon_img));

23.str1=strcat(num2str(bit_size),bit per block');

24.str2=strcat('PSNR='num2str(psnr_val));

25.final_str=strcat(str1,'—',str2);

26.title(final_str);

27.final_psnr(1)=psnr_val;
```

### 6.3. Throughput

In this section we investigate how the download throughput limitation influences the YouTube video streaming. The upload throughput in the experiments was fixed and set to 10Mbit/s. Figure 7(a) shows the dependency between the SR and download throughput ranging from 256 Kbit/s up to 10Mbit/s. We can observe that the 1Mbit/s download throughput is sufficient for our streamed video independently of the playing strategy used. Increasing the throughput beyond 1Mbit/s does not significantly improve the SR. From the other side, even a small throttle of the network throughput results in a dramatic rise of the SR value. Taking into account that the video encoding rate is 850 Kbit/s, such streaming behavior is common sense and the network throughput below this threshold value should be insufficient. Furthermore, the mentioned

threshold will be additionally increased by network protocols overhead. The similar situation is when we used the stalls number measure, Fig. 7(b). For the 1Mbit/s network throughput and above the number of stalls is 1 for the Flash and HTML5 strategies. The Simple playing strategy experiences two re-buffering events.




6.3 (a) stalling ratio;(b) re-buffering events

### 6.4. The influence of throughput limitation.

Generally, we can conclude that the all three playing strategies cannot satisfactorily cope with even small limitation in the network throughput, and the initial buffering implemented by the Flash and HTML5 strategies fails in this situation. Youtube dynamically changes the quality of video during playback. Even though that transition may appear seamless to you, this interrupts your local caching because it's a different bit rate and resolution video. And this may happen multiple time over the course of playback, each time interrupting your local cache. So for example, a video may start playing at 360P, but when the system detects that your bandwidth can handle additional 720p, it will transition over to 720p. When that happens, it has to write a new cache file, because you cannot have a video file with two different resolutions. So when you try

to go back to time period before the resolution-transition took place, your existing local cache gets wiped, and new cache starts. Another common reason is, people often fast-forward when they watch videos, which also interrupts local caching. Let's say you are 5 minutes into a video, but you've skipped the first 30 seconds. At this point, Youtube  has cached 0:30 - 5:00, so you are free to drag your timeline within that spot. But if you go to 0:00 - 0:30,
your player has to download that stream, and 0:30 - 5:00 cache will be lost. With that said, we can avoid the cache interruption problem by doing the followings:
As soon as video begins, change the Video quality setting to your desired resolution (360p, 480p, 720p, etc.) and turn off the Auto. And don't fast-forward. By taking these steps, you'll ensure that the video is cached locally in its entirety, and you'll be able to move back and forth within the timeline without reloading.

# Chapter 7

## Conclusions

This reports presents the answer how network defects, manifested as latency, packet loss and throughput limitations and impact the quality of HTTP based video playback. For this purpose, an experimental evaluation of YouTube video transmission has been conducted to examine the quality of experience of end user applications expressed as a function of playback buffer occupancy.

Through this analysis it is investigated that how long the video playback is stalled and how often re-buffering events take place. Generally, in order to watch the 850 Kbit/s video without interruptions and extensive buffering time, the packet delay introduced by the network should not exceed 200ms. The packet loss higher than the 0.8% makes the viewing of online video very inconvenient. For smooth transmission of the video, network connection throughput should be at least 1Mbit/s. The analysis reveals that there exist some small differences between the Flash and HTML5 strategies; however, in the most cases they will remain unnoticed by the end user. The buffering algorithm used in an HTML5 player showed the highest resilience against the packet delay and loss when taking into account the number of re-buffering events experienced during the video play. However, when comparing these both strategies with the Simple strategy, it is obvious that starting the video playback as soon as a minimum buffer level is achieved is insufficient. Although the Simple strategy has lower stalling time compared to the other two strategies, nonetheless, the number of re-buffering events which occur during the video streaming in unacceptable for an end user. All the three playing strategies cannot satisfactorily cope with even small limitation in the network throughput. The initial buffering mechanism implemented by the Flash and HTML5 strategies fails in that situation.

# References

[1] “Global mobile data traffic forecast update, 2010002015”, *Cisco Vi- seal Networking Index. White Paper*, Cisco, 2011.

[2] A. Rae, Y. S. Lim, C. Baraka, A. Logout, D. Toweled, and W. Dubious, “Network characteristics of video streaming traffic”, in *Proc. 7th Int. Conf. Emerge. Newt. Expert. Technol. Context 2011,* Tokyo, Japan, 2011.

[3]. Maier, G., Feldman, A., Parson, V., Allan, M.: On dominant characteristics of residential broadband internet traffic. In: Proc. of 9th ACM SIGCOMM Internet Measurement Conference (IMC 2009), Chicago, Illinois, USA (November 2009)

[4]. Kontothannsis, L.: Content Delivery Consideration for Web Video. In: Keynote at ACM Multimedia Systems 2012 (MMSys 2012), Chapel Hill, North Carolina, USA (2012)

[5]. Brodersen, A., Scellato, S.Wattenhoefer, M.: YouTube Around the World: Geographic Popularity
of Videos. In: Proc. of 21th International World Wide Web Conference (WWW2012), Lyon, France (April 2012)

[6]. Ghobadi,M., Cheng, Y., Jain, A.MatthisM.: Trickle: Rate Limiting Youtube Video Streaming.
In: Proc. of 2012 USENIX Annual Technical Conference, Boston, MA, USA (June 2012)

[7]. Adhikari, V.K., Jain, S., Chen, Y., Zhang, Z.L.: Vivisecting YouTube: An Active Measurement
Study. In: Proc. of IEEE INFOCOM 2012 Mini-Conference, Orlando, Florida, USA (March 2012)

[8]. Adhikari, V.K., Jain, S., Zhang, Z.L.: YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective. In: Proc. of 10th ACM SIGCOMM Internet Measurement Conference (IMC 2010), Melbourne, Australia (November 2010)

[9]. Adhikari, V.K., Jain, S., Zhang, Z.L.: Where Do You “Tube”? Uncovering YouTube Server

Selection Strategy. In: Proc. of International Conference on Computer Communications and Networks (ICCCN 2011), Maui, Hawaii, USA (August 2011)

[10]. Finamore, A., Mellia, M., Munafo, M., Torres, R., Rao, S.: YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In: Proc. of 2011 ACMSIGCOMM Internet Measurement Conference (IMC 2011), Berlin, Germany (November 2011)

[11]. Plissonneau, L., Biersack, E.: A Longitudinal View of HTTP Video Streaming Performance. In: Proc. of ACM Multimedia Systems 2012 (MMSys 2012), Chapel Hill, North Carolina, USA (February 2012)

[12]. Torres, R., Finamore, A., Kim, J.R., Mellia, M., Munafo, M.M., Rao, S.: Dissecting video server selection strategies in the youtube cdn. In: Proc. of 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapolis, Minnesota, USA (June 2011)

[13]. Zhou, J., Li, Y., Adhikari, V.K., Zhang, Z.L.: Counting YouTube videos via random prefix sampling. In: Proc. of 2011 ACM SIGCOMM Internet Measurement Conference (IMC 2011), Berlin, Germany (November 2011)

[14]. Rao, A., Legout, A., Lim, Y., Towsley, D., Barakat, C., Dabbous,W.: Network characteristics of video streaming traffic. In: Proc. of 7th International COnference on emerging Networking EXperiments and Technologies (CoNEXT 2011), Tokyo, Japan (December 2011)

[15]. Plissonneau, L., Bier sack, E., Juluri, P.: Analyzing the Impact of YouTube Delivery Policies on the User Experience. In: Proc. of 24th International Tele traffic Congress (ITC'24), Krakow, Poland (September 2012)

[16]. Alcock, S., Nelson, R.: Application flow control in YouTube video streams. ACM SIGCOMM Computer Communication Review 41(2) (April 2011)

[17]. Hoßfeld, T., Schatz, R., Biedermann, S., Platzer, A., Egger, S., Fiedler, M.: The Memory Effect and Its Implications on Web QoE Modeling. In: Proc. of 23rd International Tele traffic Congress (ITC'23), San Francisco, USA (September 2011)

[18] S. Alcock and R. Nelson, "Application flow control in YouTube video streams". *ACM SIGCOMM Comp. Commun. Rev.*, vol. 41, no. 2, pp. 24–30, 2011.

[19] I. Hickson, "HTML5: a vocabulary and associated APIs for HTML And XHTML", April 2010 [Online]. Available: http://www. w3.org/TR/html5/

[20] A. Finamore, M. Mellia, M. Munafo, R. Torres, and S. R. Rao, "YouTube everywhere: impact of device and infrastructure synergies on user experience", Tech. Rep. 418, Purdue University, May 2011.
[21] S. Acharya and B. C Smith, "Experiment to characterize videos stored on the web", in *Proc. ACM/SPIE Multimedia Comput. Netw. MMCN 1997*, vol. 3310, pp. 166–178, 1997.
[22] S. Acharya, B. Smith, and P. Parns,"Characterizing user access to video on the world wide web", in *Proc. ACM/SPIE Multimedia Com-*
*put. Netw. MMCN 2000*, vol. 3969, pp. 130–141, 2000.
[23] R. Rejaie, M. Handley, and D. Estrin, "Quality adaptation for congestion
controlled video playback over the internet", SIGCOMM
Comput. Commun. Rev., vol. 29, no. 4, pp. 189–200, 1999.
[24] S. Floyd, M. Handley, J. Padhye, and JÃ¶rg Widmer, "Equationbased
congestion control for unicast applications", *SIGCOMM Com-*
*put. Commun. Rev.*, vol. 30, no. 4, pp. 43–56, 2000.
[25] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization:
a view from the edge". in *Proc. 7th ACM SIGCOMM Conf.*
*Internet Measur. IMC 2007*, San Diego, CA, USA, 2007, pp. 15–28.
[26] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube
network traffic at a campus network-measurements, models, and implications", *Computer Netw.*, vol. 53, no. 4, pp. 501–514, 2009.
[27] L. Plissonneau, T. En-Najjary, and G. Urvoy-Keller, "Revisiting web
traffic from a DSL provider perspective: the case of YouTube", in
*Proc. ITC Spec. Seminar Netw. Usage Traffic*, Berlin, Germany, 2008.
[28] M. Cha, H. Kwak, P. Rodriguez, Y. Y Ahn, and S. Moon, "I tube,
you tube, everybody tubes: analyzing the world's largest user generated
content video system", in *Proc. 7th ACM SIGCOMM Conf.*
*Internet Measur. IMC 2007*, San Diego, CA, USA, 2007, pp. 1–14,
2007.
[29] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos", in *Proc. 16th Int. Worksh. Quality of Service IWQoS 2008*, Enschede, The Netherlands, 2008, pp. 229–238.

[30] A. Abhari and M. Soraya, “Workload generation for YouTube”, *Mul-timedia Tools and Appl.*, vol. 46, no. 1, pp. 91–118, 2010.

[31] D. K. Krishnappa, S. Khemmarat, and M. Zink, “Planet YouTube: global, measurement-based performance analysis of viewer;’s experience watching user generated videos”, in *Proc. IEEE 36th Conf. Local Comp. Netw. LCN 2011*, Bonn, Germany, 2011, pp. 948–956.

[32] S. Benno, J. O. Esteban, and I. Rimac, “Adaptive streaming: the network HAS to help”, *Bell Labs Tech. J.*, vol. 16, no. 2, pp. 101–114,2011.