Video Stream Monitoring and Network-centric QoE Prediction through User-behavioral Studies and Automated Learning

DHANANJAYA KUMARA KITTUR GONIBASAPPA
Video Stream Monitoring and Network-centric QoE Prediction through User-behavioral Studies and Automated Learning

Dhananjaya Kumara Kittur Gonibasappa
August 2017

This project has been done at Telekom Innovation Laboratories Berlin, in collaboration with Technische Universität Ilmenau, Germany

Examiner: Prof. Dr. Ahmed Hemani
School of ICT
KTH Royal Institute of Technology Stockholm, Sweden

Supervisors: Dimitrios Stathis
School of ICT
KTH Royal Institute of Technology Stockholm, Sweden

Werner Robitza
Telekom Innovation Laboratories Berlin, Germany

Prof. Dr. Alexander Raake
Technische Universität Ilmenau, Germany

Dr.-Ing. Bernhard Feiten
Telekom Innovation Laboratories Berlin, Germany
Acknowledgement

First and foremost I would like to thank my thesis supervisor Werner Robitza at Telekom Innovation Laboratories for his ideas, analysis and constant feedback throughout my work.

I would like to thank my thesis advisers, Prof. Dr. Alexander Raake and Dr.-Ing. Bernhard Feiten at Telekom Innovation Laboratories and TU-Ilmenau for giving me this opportunity, guidance, facilities, and support needed to complete this thesis. I wish to express my sincere gratitude to Alexander Dethof, Ulf Wüstenhagen and Steve Göring at Telekom Innovation Laboratories and TU-Ilmenau for providing necessary guidance throughout the project. I thank all of them for supporting my ideas and working patiently with me.

I also would like to thank Prof. Dr. Ahmed Hemani and Dimitrios Stathis at KTH Royal Institute of Technology for offering me the support, guidance, and feedback regarding my work. I am grateful for them to giving me the time regardless of their busy schedule.

I am always thankful to my family for their love and support. This accomplishment would not have possible without them.
Video Stream Monitoring and Network-centric QoE Prediction through User-behavioral Studies and Automated Learning

Dhananjaya Kumara Kittur Gonibasappa

November 20, 2017

Abstract

Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service [1]. To ensure a proper level of QoE for end users, networks and service providers have to continuously monitor their systems in terms of technical parameters, which can then be used to estimate QoE. Especially for video streaming services, which consume a large amount of traffic, network problems such as bandwidth fluctuations quickly develop into annoying artefacts visible to the users, which may lead to abandonment of services. Internet Service Providers (ISPs) are therefore continuously monitoring video network streams in order to provide the better QoE. In this regard to conduct the user behavioral studies, the ISPs spend a large amount of money and energy every time. To avoid this, we are using existing user behavioral studies and simulating the user behavior in an automated set-up and try to measure the impact of network conditions.

In our current studies based on the user-behavioral model used [5], we can conclude that low upload speeds don’t affect on simulated user behavior unless they are in high download speed networks. Simulated users with the mid-range download and upload bandwidth tend to face more stalling and quality switches compared to both low and high-bandwidth users. Key quality indicators (KQIs) of video QoE also depends on the number of videos we measure in a single session. Reloading of player helps to reduce stalling for mid and high bandwidths. Reloading worsens the situation in low bandwidth scenarios.

Keywords - Quality of Experience, Video streaming, User behavioral studies, Video monitoring, Key quality indicators v/s Network conditions, Automated learning.
Sammanfattning

Kvalitet av erfarenhet (QoE) definieras som: "Graden av fröjd eller förargelse av användaren av en applikation eller en service. Den resulterar från uppfyllelsen av hans eller hennes förväntningar med hänsyn till hjälpmedlet, och/eller njutning av applikationen eller servicen i ljuset av användarens personlighet och aktuella tillstånd" [1]. Att se till en riktig nivå av QoE för slut - användare, nätverk och tjänste- familjeförsörjare måste övervakar fortlöpande deras system när det gäller tekniska parametrar, som kan därefter vara den van vid bedömningen QoE. Speciellt för videoen som strömmar service, som konsumerar ett stort belopp av trafik, framkallar nätverksproblem liksom bandbreddväxlingar snabbt in i förargliga artefacts som är synliga till användarena, som kan leda till övergivande av service. Internetleverantörer (ISPs) är därför fortlöpande videopp nätverksströmmar för övervakning för att ge den bättre QoEen. I detta avseende att föra de beteende- studierna för användaren, spenderar ISPsna en stor mängd pengar och energi varje gång. Att undvika denna, använder simulerar vi beteende- studier för existerande användare och användareuppförandet i en automatiserat aktivering och försöker för att mäta inverkan av nätverksvillkor.


**Nykkelord** - Kvalitet av erfarenhet (QoE), läggande tillbaka video som strömmar, beteende- studier för användare, video övervakning, Nyckel- kvalitets- indikatorer v/s Nätverksvillkor, automatiserat lära.
# Contents

Abstract i

Contents viii

List of figures x

List of tables xi

1 Introduction 1
   1.1 Domain Problem: Measuring the user behavior with respect to different network conditions 2
   1.2 Research Questions 3
   1.3 Methodology 3
   1.4 Benefits, Ethics, and Sustainability 5
   1.5 Summary 5
   1.6 Thesis Outline 6

2 Technical Background 7
   2.1 Introduction to Streaming Video 7
   2.2 Quality of Experience 8
   2.3 Measuring Quality of Experience 8
   2.4 Subjective Quality 9
   2.5 Instrumental Quality 9
   2.6 Quality Models 10
      2.6.1 Classification Quality Models 10
   2.7 User Behavior 11
   2.8 User Engagement 11
   2.9 Quality Metrics 12
      2.9.1 Initial Loading Delay 12
      2.9.2 Average Stalling Duration 12
      2.9.3 Number of Stalling Events 12
      2.9.4 Number of Quality Switches 12
      2.9.5 Total Stalling Duration 12
      2.9.6 Total Stalling Ratio 12
      2.9.7 Stalling Ratio 13
      2.9.8 User Abandonment Rate/Abandonment Rate 13
   2.10 YouTube Architecture 13
      2.10.1 How Google’s CDN Work 14

3 Project Background 15
### CONTENTS

3.1 User Behavioural Studies .................................................. 15  
3.2 Motivation ........................................................................ 16  
3.2.1 Overview of Large-Scale User-Behavioral QoE Monitoring ..... 17  
3.2.2 Proposed Solution .......................................................... 17  
3.3 Methodologies ................................................................... 18  
3.3.1 User-Behavioral Model .................................................... 19  
3.3.2 Different User-Behaviors ............................................... 21  

4 Implementation ..................................................................... 25  
4.1 Architectural Diagram of the Implementations ....................... 25  
4.2 Network Module .................................................................. 25  
4.2.1 Wondershaper ............................................................... 26  
4.2.2 Speedtest ................................................................. 27  
4.2.3 Python Script ............................................................ 30  
4.3 Server Module ................................................................... 30  
4.4 Client Module .................................................................... 32  
4.4.1 Generation of the Automated User-Behaviour .................. 33  
4.4.2 Implementation of YouBot ............................................. 33  
4.4.3 Youtube Iframe APIs .................................................... 35  
4.5 Data Analysis Module .......................................................... 38  
4.5.1 Data Selection and Aggregation ....................................... 39  
4.5.2 Plots ............................................................................ 39  
4.5.3 Automated Learning Model .......................................... 39  
4.6 Impression of the Data Collected ......................................... 39  
4.6.1 Console logs ................................................................ 40  
4.6.2 Network logs ............................................................. 41  
4.6.3 Output of Calculated KQIs ........................................... 41  
4.7 Hardware Used .................................................................. 43  

5 Results and Analysis ............................................................... 45  
5.1 Impact of Network Conditions on KQIs ................................. 45  
5.1.1 Initial Loading Delay .................................................... 45  
5.1.2 Average Stalling Duration .............................................. 45  
5.1.3 Total Stalling Duration .................................................. 46  
5.1.4 Number of Stalling ....................................................... 48  
5.1.5 Number of Quality Switches ......................................... 48  
5.1.6 Total Stalling Ratio ....................................................... 50  
5.1.7 Stalling Ratio ............................................................. 50  
5.1.8 User Abandonment Rate .............................................. 51  
5.2 Impact of Measuring Single video session compared to the Multiple video session on KQIs ......................................................... 53
List of Figures

2.1 A 60-second video with quality variations and stalling in the middle . 8
2.2 Video player events and corresponding KQIs ................................. 13
3.1 Large-scale user-behavioral studies ............................................ 18
3.2 Proposed solution to measure the impact of network conditions ... 19
3.3 A. Correlation between percentage of abandon view and startup delay
   B. Simple regression fitting for abandonment rate v/s startup delay [5] 20
3.4 Inverse regression fitting for abandonment rate v/s initial loading delay 21
3.5 Flow diagram for Case 1 ............................................................ 22
3.6 Flow diagram for Case 2 ............................................................ 23
3.7 Flow diagram for Case 3 ............................................................ 24
4.1 Architectural diagram of the implementation ................................ 26
4.2 Block diagram explaining tc stack [42] ......................................... 27
4.3 Speed test using speedtest.t-online.de ...................................... 27
4.4 VDSL connection comparison .................................................... 28
4.5 Diagram showing the average results ......................................... 29
4.6 Ping test using the Linux ping cmd ............................................. 29
4.7 Speed test using the python lib Speedtest ................................... 30
4.8 UML of the active video probe .................................................. 31
4.9 Architectural diagram of the user behavioral simulation ............... 34
4.10 YouBot class diagram .............................................................. 36
4.11 Screen shot showing the video playing in the YouTube ............... 40
4.12 Screen shot showing the network events in the browser ............... 42
5.1 Impact of upload/download speed on initial loading (startup) delay . 46
5.2 Impact of upload/download speed on average stalling duration ....... 47
5.3 Impact of upload/download speed on total stalling duration .......... 47
5.4 Impact of upload/download speed on number of stalling .......... 49
5.5 Impact of upload/download speed on number of quality switches ... 49
5.6 Impact of upload/download speed on total stalling(buffering) ratio . 51
5.7 Impact of upload/download speed on stalling(re-buffering) ratio .... 52
5.8 Impact of upload/download speed on user abandonment rate ....... 52
5.9 Comparison of case 1 and 2 on initial loading delay ..................... 54
5.10 Comparison of case 1 and 2 on average stalling duration .......... 54
5.11 Comparison of case 1 and 2 on total stalling duration .......... 55
5.12 Comparison of case 1 and 2 on number of stalling .................. 56
5.13 Comparison of case 1 and case 2 on number of quality switches .... 57
5.14 Comparison of case 1 and 2 on total stalling(buffering) ratio .... 58
5.15 Comparison of case 1 and 2 on stalling (re-buffering) Ratio ....... 58
5.16 Comparison of case 1 and 2 on user abandonment rate ............ 59
5.17 Comparison of case 2 and 3 on initial loading delay ................ 60
LIST OF FIGURES

5.18 Comparison of case 2 and 3 on average stalling duration ............... 61
5.19 Comparison of case 2 and 3 on total stalling duration .................... 61
5.20 Comparison of case 2 and 3 on number of stalling .......................... 62
5.21 Comparison of case 2 and 3 on number of quality switches ................. 63
5.22 Comparison of case 2 and 3 on total stalling (buffering) ratio ............ 63
5.23 Comparison of case 2 and 3 on stalling (re-buffering) ratio ............... 64
5.24 Comparison of case 2 and 3 on user abandonment rate ...................... 65
5.25 Percentage of videos loaded after initial loading delay < 5s ............... 65
5.26 Percentage of videos loaded after initial loading delay < 5s after first reload .................................................. 66
5.27 Percentage of videos loaded after initial loading delay < 10 s after second reload .................................................. 67
5.28 Percentage of videos loaded after initial loading delay > 10 s even after second reload ........................................ 67
5.29 Curve fitting for initial loading (startup) delay .............................. 71
5.30 Curve fitting for average stalling duration ................................... 75
5.31 Curve fitting for total stalling duration ..................................... 78
A.1 Box plots showing the impact of upload speed on initial loading delay ... 90
A.2 Box plots showing the impact of download speed on average stalling duration .................................................. 93
A.3 Bar plots showing the total time at each quality levels for different upload speeds ........................................ 95
A.4 Bar plots showing the total time at each quality levels each download speeds ........................................ 98
A.5 Comparison of case 1 and 2 showing the total time at each quality levels ........................................ 99
A.6 Comparison of case 1 and 2 showing the total time at each quality levels ........................................ 99
A.7 Comparison of case 2 and 3 showing the total time at each quality levels ........................................ 100
A.8 Comparison of case 2 and 3 showing the total time at each quality levels ........................................ 100
List of Tables

1. One to one mapping between user actions and automated actions. . . . 33
2. Player states [48] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
3. Player quality levels [48]. . . . . . . . . . . . . . . . . . . . . . . . . . 37
4. Technical specification of the embedded hardware used. . . . . . . . 43
5. Table of coefficients for initial loading delay . . . . . . . . . . . . . . 68
6. Results for initial loading delay . . . . . . . . . . . . . . . . . . . . . 70
7. Table of coefficients for average stalling duration . . . . . . . . . . . 72
8. Results for average stalling duration . . . . . . . . . . . . . . . . . . . 74
9. Table of coefficients for total stalling duration . . . . . . . . . . . . . 74
10. Results for total stalling duration . . . . . . . . . . . . . . . . . . . . 77
1 Introduction

In today’s digital technology era, all entertainment fields are advancing including video streaming services. The amount of video on the internet accounts for 82 percent of all consumer internet traffic by 2021, up from 73 percent in 2016 [2]. The continuous advancement in the internet technology and bit-rates have increased the user expectation towards quality and resolution of the video they watch in recent days. In 2000 anything more than Standard Definition (SD, 640 × 480) was considered to be good by users, but today even Ultra High Definition (UHD, 3840 × 2160) seems to be common on all screens and devices. This behavior has made Over The Top (OTT) video streaming and content delivery networks a highly competitive landscape. To keep up with user expectations and gain more subscribers, both OTT video streaming providers and Internet Service Providers (ISPs) have to consider not only Quality of Service (QoS) of the video stream but also Quality of Experience (QoE) of the users.

OTT services are growing fast every day, they are expected to overtake Television (TV) within next five years [3]. To make their services more desirable and attract more viewers, it is important to understand the opinion of the users about their video services. Generally, user experience towards the provider’s service makes a major impact on new subscriptions for the provider. The video providers want less user abandonment, more user engagement and more re-visiters. To achieve this, they expect their videos to not to fail, start-up quickly and play without any interruptions in the middle. To understand user experience towards the video service, it is more important to understand the user behavior during the usage of the service. It is an imminent challenge for the internet video ecosystems i.e. content providers, content delivery networks and ISPs to measure the Quality of Experience from the perspective of the users.

A lot of studies focusing on viewer behavior have been conducted in the past to match its impact on QoE matrices [5]. Most of the studies concentrate on the subjective QoE [4] to calculate the Key Performance Indicators (KPIs) called Key Quality Indicators (KQIs) that can be used in the calculation of Mean Opinion Scores (MOS). One of the key questions for the internet service providers is to what extent can these KQIs affect the user behavior? In the meantime to find the most cost-effective way to make these KQIs better, so that they can achieve less user abandonment and more user engagement. All the studies which are on video QoE concentrated mostly on matching user behavior with KQIs. In this thesis, there is an effort in understanding to what extent changes in the network conditions can affect the user behavior of the viewers. We are hoping to achieve this by measuring different video KQIs with respect to the simulated user behavior and relating those to the network conditions applied during those measurements.
1 INTRODUCTION

1.1 Domain Problem: Measuring the user behavior with respect to different network conditions

In order to determine how the network quality impact user behavior, a video QoE solution needs to measure the video quality of experience from the perspective of the end users. The fundamental motivation of this project is that video service providers want to measure the impact of network conditions on end user’s experience i.e., the subscriber quality of experience for streaming video, or the video QoE. Thus, it is critical to systematically understand the interplay between network conditions, video quality, and user engagement. This knowledge could be useful to get the insight on network conditions that affect user engagement. This result or analysis can later be useful for video service providers to invest their network and server resources to optimise the quality metrics that really matter. Some of the points which need to be considered while doing the measurements are:

1. The quality must be measured with respect to the client side from the user perspective.

2. These measurements are often very cost intensive and requires a lot of human subjects getting involved, which often get ethically complex.

3. To conduct different network condition based measurements, ISPs need to intentionally degrade network conditions. ISPs can not intentionally degrade the network conditions, so this is not an adequate way to monitor QoE.

4. There are legal and ethical issues if ISPs try to intentionally degrade the stream quality of a set of viewers to conduct a controlled experiment.

5. Often this kind of measurements needs to be done in large quantities to come to a conclusion and they get cost intensive and legally complicated.

To address all the above concerns, we have designed an automated experimental set-up which emulates different network conditions and uses an existing large-scale user-behavioral study from the literature [5] and assumes/simulates user behavior on the client machine with which we measure the video QoE (More details in Section 3.3.2).
1 INTRODUCTION

1.2 Research Questions

The research questions that are addressed in this thesis are:

1. How different network conditions (i.e., upload and download speed of the network) affect different video KQIs which later, in turn, affect the simulated user engagement?

2. What is the trade-off between network conditions and simulated user-behavior on the perceived video QoE?

3. To what extent does reloading of the web-page/video player helps the simulated users in getting lower video load time, in case of the bad network condition?

4. How does simulated user abandonment rate varies with respect to a single video viewing session, compared to multiple video viewing sessions?

5. How measuring the video KQIs vary with respect to a single viewing session compared to multiple video viewing sessions?

The limitation of this work is that it adopts user-behavior model discussed in [5] and any changes to this has to be reflected also in the automated user simulation. Also, this work is conducted on TINO embedded PC running Linux mint OS using Chrome browser, and YouTube as the OTT service. Any changes to this also need to be taken into consideration.

1.3 Methodology

In this thesis, we have used quantitative approach and experimental research strategy. We wrote a Chrome extension called YouBot (Automated YouTube Bot) which simulates user behavior like starting a web browser, navigating to the user’s YouTube homepage and clicking on the most recently recommended video on the homepage. The simulated user (from this, in the document referred as the ‘user’) watches the video for 60 seconds of viewing duration and later simulated user may wish to stop watching the video or may continue to watch the next video (passive watching). If the simulated user faces more video loading time in a particular video starting up they may choose to reload/refresh the web-page (interactive watching). YouBot will simulates each behavior and start measuring the video parameters and create logs in client side. These logs are transmitted to the server by web socket connections. Once these logs are collected, an active probing script will calculate the video KQIs using these logs and save it to the file systems. Python script in server reads the KQIs form file system and do the data analysis.
1 INTRODUCTION

We have conducted the above experiment for 21 independent variable combinations which correspond to different network conditions and user-behavioral scenarios along with 8 different KQIs as dependent variables. Each experiment is repeated 21 times, resulting in approximately 15000 videos, contributing to a total duration of 250 hours of viewing time.

1. Download speeds in Mbps: 0.256, 0.384, 0.512, 0.768, 1.024, 1.536, 2.048, 3.072, 4.096, 8.192, 16.384, 32.768 and 37.499

2. Upload speeds in Mbps: 0.256, 0.512, 1.024, 2.048 and 3.379

3. User Behavioral scenarios:

   (a) Case 1: Navigate to YouTube homepage and watch a single video for the viewing duration of 60 seconds and quit the browser. It is referred now onwards as the passive single viewing session or Case 1.

   (b) Case 2: Navigate to YouTube homepage and start watching five consecutive videos by clicking on the next video after every 60 seconds of the viewing duration. It is referred now onwards as the passive multiple viewing session or Case 2.

   (c) Case 3: Navigate to YouTube homepage and start watching five consecutive videos by clicking on the next video after every 60 seconds of the viewing duration. If the user finds 5 second delay in video start-up during any video (approximately 25% of the user abandonment rate [5]) they will choose to reload the web-page/player. Once, after reloading the player measurement get started again and if second time the user still faces video start-up time of more than 5 seconds, this time they will wait till 10 seconds (approximately 50% of the user abandonment rate) and reloads again. After second reload also a new measurement is started and later we calculate by how much percent does the user got better results in each reload operation compared to the previous session. It is referred now onwards as the active multiple viewing session or Case 3.

Outliers in the data have been removed using interquartile range and analyzed the data using box plots to create confidence level interval plots for all the dependent variables. Using above data, we were able to compare all three use cases. Later we draw conclusions based on these curves. After that, we build an automated model using non-linear curve fitting. The prediction efficiency of the model is analyzed using error measures namely, PCC(Pearson correlation coefficient) and RMSE(Root Mean Squared Error).
1.4 Benefits, Ethics, and Sustainability

The social and Ethical aspects of this thesis are beneficial because:

- Service providers get the analysis they needed without getting much human intervention.

- Avoiding human test subjects is a very big plus for the ethical aspect of the research.

- Making human test subjects watch a very low-quality video to perform the experiment also seems to be ethically complex and we are ensuring this doesn’t happen with automation.

- No personal data will be collected during the experiments which in turn resolves the privacy issues.

- Degrading the network quality of the user endpoint is also avoided which in turn resolves the legal issues with broadcasting.

- Conducting automated experiments reduces the cost and efforts in repeating the new studies based the same or related parameters.

The environmental aspects of the project are comparatively insignificant as we are using a fanless TINO embedded fan-less systems[7] which produces very low heat with very low power consumption.

1.5 Summary

We tried to make a formal relation between user-behavior and different network conditions. Conducting experiments to measure different user-behavior by varying network conditions also became very simple and cost-effective through this framework. Overall, we can summarize this project with below points:

- Lower upload speeds during the high download speeds have high impact on all KQIs.

- In download speeds from 0.3 to 2 Mbps, there is a high variation in the number of stalling and quality count. This may be due to the player adoption strategy.

- Measuring the multiple video session compared to the single video session has an impact on KQIs and during the multiple video session we get better KQI values compare to single video session.
1 INTRODUCTION

- Reloading only reduces the startup delay in mid-range upload and download speeds. Reloading in lower bandwidth increase the initial loading delay.

- Reloading player helps for mid and high range network bandwidths.

- Quality switching happens more often in mid-range download and upload speeds due to network adaptation strategy of YouTube Player.

- In case of low bandwidths, it took more time for the video to load, and once started, it plays without stalling.

- Network conditions alone do not directly affect the user-behavior as some of the KQIs have bad values for mid-range download bandwidth compared to lower bandwidths.

1.6 Thesis Outline

The organization of this thesis report is as follows.

- Chapter 2: The necessary technical background to understand the thesis is given here.

- Chapter 3: Project background, literature review and motivation for the thesis is discussed here.

- Chapter 4: A detailed description of the project background, tools used, implementation of software and outline of measurements conducted are explained here.

- Chapter 5: The results from the measurements are interpreted and analyzed with respect to the different simulated user-behavior and automated model creation are discussed here.

- Chapter 6: The outcomes of the thesis, conclusion, future work and the lessons learned during the thesis are discussed here.
2  Technical Background

This chapter provides the technical background and literature study done in order to understand this thesis in depth. Here we have mainly discussed the streaming video, different types of the streaming video technologies, basic definitions of quality of experience (QoE), different video quality models, and why it is important to measure the user behavior with respect to different network conditions.

2.1 Introduction to Streaming Video

Streaming video is defined as watching a video in real time as it gets downloaded from the internet. Using streaming technology, users can watch the video while it is being transmitted to their browser instead of waiting for the complete video to download and then playing it.

Usually, video files require large amount of disk space, for example, a five-minute uncompressed video may take around 2GB of disk space. For this reason, when video files are prepared for online streaming, it is compressed to make the file size smaller. When a user requests the video, the compressed video is sent from a video server, and it is decompressed by a video player on the client’s browser to play it in real time. A user can move to any point in the video length and start watching it. The streaming video tries to keep pace with the user’s network speed in order to reduce interruptions and stalling [8].

There are two main types of video streaming over Hypertext Transfer Protocol (HTTP):

- Progressive download.
- HTTP adaptive streaming (HAS).

A detailed information on progressive download and HTTP adaptive streaming can be found in [9]. One of the standardized HAS method YouTube uses is Dynamic Adaptive Streaming over HTTP (DASH)[10].

If the user network bandwidth is high, the adaptive client will download the best resolution video. If the bandwidth suddenly drops, the client will switch to a lower resolution video until the network conditions improve. The purpose of this is to avoid stalling, which usually gives a bad user experience [11]. As we can see in the Figure 2.1 if the initial bandwidth conditions are good, then the video starts with the high resolution i.e., 1280 × 720 pixels after a very small initial loading delay, and in the meantime, during the playout if user face some network fluctuations the video quality changes to a lower resolution which would have already downloaded with the DASH technology. After changing to a lower resolution than the user may
2  TECHNICAL BACKGROUND

Figure 2.1: A 60-second video with quality variations and stalling in the middle

face a stalling in between the video watching session and once the network conditions become better then automatically the player starts playing in high resolution.

2.2 Quality of Experience

There are many available definitions for QoE, in[9] the authors define it as 'Quality of Experience(QoE) is the degree of delight or annoyance of a person whose experiencing involves an application, service, or system. It results from the person’s evaluation of the fulfilment of his or her expectations and needs with respect to the utility and/or enjoyment in the light of the person’s context, personality, and current state'. According to the definition, video QoE is a subjective phenomenon and it also differs from user to user. This makes measuring the factors that influence video QoE a major technical challenge. Measuring the Network flow or TCP packet drops(QoS model) do not give any inference regarding the user experience. There are many studies [26] concentrated on the importance of QoE model over QoS model.

2.3 Measuring Quality of Experience

To support service providers user experience measurement initiatives, a video QoE solution must measure the video quality of experience from the perspective of the end user. Here service providers want to measure the user quality of experience
for streaming video with respect to their network conditions in order to optimise their offerings. By this definition, video QoE is distinct from network health metrics, transport layer performance (TCP packet drops, round-trip time, jitter), bandwidth averages, etc.

Generally, there are two major types of measuring the QoE namely subjective and instrumental quality assessment.

### 2.4 Subjective Quality

Subjective video quality is a way of measuring video quality by showing videos to a group of users called subjects in the lab equipped with Computers or TVs in a controlled environment. This involves the subjects rating videos based on their experience of watching a particular video. These videos can be presented to subjects in isolation (absolute rating) or in pairs. In paired comparison quality is measured with respect to a reference video. For absolute category ratings (ACR) tests, ITU-T Rec. P.910 [12] has defined a 5-point ACR quality scale which is also called as MOS-scale. Subjects are asked to judge the quality of video according to the MOS scale from 1 to 5, where 1 is bad quality and 5 being excellent quality. Quality ratings obtained from the subjects are then averaged to get the Mean Opinion Score (MOS). The subjective quality approach requires a balanced set of a sufficient number of subjects that represent a different level of expertise, age groups, and gender. One of the major disadvantages of the subjective quality tests is they are very difficult to scale to do large-scale measurements. One of the ways is that to overcome this is by doing instrumental quality tests.

### 2.5 Instrumental Quality

In order to estimate the perceived quality, mathematical models are used to approximate the results from the subjective video quality tests. Here the model represents a statistical model in which several independent variables like packet loss of bandwidth, signal to noise ratio and bit error rates are fit against the results obtained in the subjective quality test using the regression techniques. If the instrumental model is trained from the data obtained in the subjective quality model it is called an objective video quality models. Objective video quality models are based on the availability of the original video signal at the measuring end [9]. In [13] an objective video quality model for digital cable television is discussed where the full reference of the original signal is available.
2.6 Quality Models

There are many quality models developed till date with respect to the different requirements and criteria. Some of the QoE models are based on the applications and others are generic. QoE models are developed for specific applications such as cable television, IP television, web browsing, Over the top (OTT) video services and online gaming. Video quality models use video quality estimators such as PSNR, MSE, Packet loss, and more complex signal processing techniques.

2.6.1 Classification Quality Models

Quality model classification can be done based on different criteria[14].

1. Based on the amount of information available from original signal[14]:
   
   (a) Full Reference Methods (FR): Here the original video signal is compared against the received video signal after the transmission. Each pixel is compared with the corresponding pixel. Generally, these models will not use any information about the type of video encoding or transmission process involved. FR method gives higher accuracy with more computational effort compared to the other methods.

   (b) Reduced Reference Methods (RR): Here the original video signal is not fully available but some features from both original and received video signals are compared each other. In RR method whole original video is not required to transmit hence makes it more computationally efficient than FR metrics.

   (c) No-Reference Methods (NR): Here the received video signal is used to calculate the quality without using any reference to the original video signal. The prediction accuracy of NR methods is lower compared to FR and RR methods.

2. Based on the type of the input used by the model[14] [15]:
   
   (a) Planning Model: Here models are designed based on the service information available during the network planning phase [15].

   (b) Parametric Model: Here models are designed based on the information extracted from packet headers during the measurement [15].

   (c) Bit Stream Model: Here models are designed based on the bit-stream level information available during the measurement [15]. According to [15] there are two types of bit-stream level models i.e., Mode 1 where the model does not fully decode the payload information, but only parses the
bit-stream and Mode 2 where the model can fully decode the bit-stream information.

(d) Hybrid Model: Here models can use information from packets headers, bit-stream along with the media information [15].

Quality models can also be classified based on service type [14](IPTV, Video on demand, mobile TV, SD, and High Definition), application for which they are developed [14](codec testing, network planning) and type of the model output [14] (MOS). Video quality models like ITU-T G.1070 [16] finds video quality based on codec type, video frame, and packet loss. QoE model for HTTP based video streaming system and scalable video coded streaming is proposed in ITU-T P.1203 [17].

2.7 User Behavior

Many objective quality models fail to give the exact imitation of the user behavior in order to determine the user-perceived quality as discussed in [5]. It is crucial to study the user behavior to make a better decision on the quality of the video which is broadcasted by the service provider and in order to do so we need to conduct subjective studies. Subjective studies are always complex and expensive to do so as they involve human subjects. This is the reason many studies tend to limit the scope to smaller user panels. Doing studies on the smaller group doesn’t indicate proper statistics due to the limitations in finding the balanced user panels. Also conducting the measurements for different geographical locations and network conditions also become more difficult in the lab environment. To overcome all these limitations, these measurements should be conducted on a large scale. In this regard, there are many studies done in large scale to do the study on the user engagement and their behavior like [24], [25] and [28].

2.8 User Engagement

User Engagement is the key parameter which has achieved more traction among the service providers in order to determine how much they can engage their users in using their services uninterruptedly or regularly. In recent days a number of studies which concentrated in calculating the user engagement for the OTT services. As an example shown in the paper [38] we can see that the user engagement or behavior studies give the deeper insight on the overall QoE of the user, as inferred from the user action.
2.9 Quality Metrics

Below is the list of KQIs or quality metrics which are used in this thesis for the analysis.

2.9.1 Initial Loading Delay

It is the duration of the time at which the player initiates a connection to a video server i.e., the user clicks on the video to till the time where sufficient amount player video buffer has filled up and the player starts rendering video frames and moves to playing state i.e., the user start seeing the video getting played. As illustrated in Figure 2.2, this is the length until the video player is in the buffering (in this thesis buffering and stalling are used interchangeably) state. Initial loading time also considered as one of the stalling events in many quality models. In this thesis, initial loading time also referred as startup delay.

2.9.2 Average Stalling Duration

It is the average duration of the stalling events that occurred during the playout of a video. In Figure 2.2 Average Stalling duration is calculated by taking the average of startup delay, stalling event 1 and stalling event 2.

2.9.3 Number of Stalling Events

It is the count of how many stalling events occurred in the total viewing duration of the video including initial loading delay. In Figure 2.2 there are 3 stalling events i.e initial loading delay, stalling event 1 and stalling event 2.

2.9.4 Number of Quality Switches

It is the number of the quality level that has been switched during the video playout. In Figure 2.2 we have total 3 quality events corresponds to the quality count of 3.

2.9.5 Total Stalling Duration

It is the sum of all the stalling events occurred in the video playout including the initial loading delay. In Figure 2.2 total stalling duration is the sum of initial loading delay, stalling event 1 and stalling event 2.

2.9.6 Total Stalling Ratio

It is the fraction of the total viewing duration (i.e playing duration + stalling) which is spent in video stalling including initial loading delay. As illustrated in Figure 2.2,
2.9.7 Stalling Ratio

It is the fraction of the total viewing duration which is spent in video stalling after initial loading delay. In Figure 2.2 we calculate the stalling ratio as the fraction of total stalling duration minus initial loading delay with respect to the total viewing duration. In this thesis, total stalling ratio is also referred as the re-buffering ratio.

2.9.8 User Abandonment Rate/Abandonment Rate

It is the rate of the user abandoning the service after waiting for the certain period of initial loading delay of the video. Percentage of abandoned views and initial loading delay are positively correlated[5].

2.10 YouTube Architecture

In this study, we have used the YouTube OTT service as the source of the videos for our analysis and monitoring. According to [18] YouTube is the second most accessed site all over the world. There are many studies conducted on YouTube traffic [21] [34] [31], its streaming behaviour [20] [22], its infrastructure [35] and Google’s Content delivery network (CDN) architecture [23]. YouTube reaches its users with different
video services by using Google’s CDN (Content delivery network). CDN is a globally distributed network of proxy servers deployed in multiple data centers and it serves content to end-users with high availability and high performance. One of the benefits of using CDN is that it replicates content in different geographical locations so that there’s a better chance of content being closer to the requested user, with fewer hops. In addition, CDN takes into consideration of issues like scalability and high availability.

2.10.1 How Google’s CDN Work

Dispute YouTube being the most popular video service choice yet we know very little about its CDN network. There are some studies like [23] dedicatedly concentrated on Google’s CDN architecture and its impact on QoE. When a user requests for a video on YouTube, it initially reaches a front-end server and looks for the presence of a particular video in its CDN network. Once it finds the requested video from the nearest CDN it looks for the shortest path to reach the requested user. A CDN has Points of Presence (PoPs) or data centers that are situated all around the world. In each PoPs there are thousands of servers and both help to accelerate the speed at which content is sent to the user.
3 | Project Background

In this chapter, we are going to discuss the literature review and motivation for this thesis. It introduces the needs for user-behavioral studies in large scale for quality of experience (QoE) monitoring. Here we briefly look into the related work done in this regard and motivation for measurement methodology used in this thesis.

3.1 User Behavioural Studies

As discussed in Section 2.7 we should measure QoE in perspective of a user. To overcome the limitations of conducting small lab controlled measurements explained in Section 2.7, we need to conduct these studies with the larger audience. In [28] and [27] large user behavioral studies on video on demand systems are studied. Especially in [27], a study in conducted with respect to the wide variety of subject and for different video lengths are done. They have shown that their results can be used to define a user-behavioral model which can be used with a service model in order to predict future situations to avoid performance problems. In [30], authors conducted the behavioral studies by introducing the time-varying distortion in video quality. In [26], authors studied the impact of the video playback events on the user video abandonment rate using YouTube. They have shown it is important to measure video playback events compared to the network events in understanding the user behavior. In their studies, it is evident that stalling (re-buffering) affects 6 times higher than initial loading delay (startup delay) when measuring the user abandonment rate, one stalling (re-buffering) event can cause 3 times more user abandonment than one bitrate change and varying bitrate can cause 4 times higher abandonment rate than constant bitrate.

In [25] authors studied user behaviors in large scale for IPTV environment using content types like Video on Demand (VoD), Digital Video Recorder(DVR), and Live TV. In [33] there is a study of user behavior in Peer-to-Peer live video streams and the authors have shown that QoE is not only user-centric but also changes with the changing situations. Our intention in the thesis is to study the user behavior with respect to the network conditions and in this regard also there are some studies done like [29], [31] and [34]. In [29] authors have studied the impact of 31 different network factors on the user-behavior in a mobile network. They have shown that the network parameters directly affect the QoE. Studies like [36], [24] and [5] shows that QoE influences the user-behavior and engagement largely. In [29] they have shown that reducing mean signal-to-interference ratio by 1dB can reduce the user abandonment ratio by 2%. This study does not use any client or server side logs as they mentioned that it is hard for a network operator to get the logs as they don’t have any infrastructure to do so. In [31] authors did a study on the impact
of HTTP streaming in YouTube traffic and found out that to adapt to changing network conditions YouTube tries to send 33% more traffic than required. Due to this nature of YouTube network operators has to allocate the network resources to video in such a way that avoids the varying network conditions. This will reduce the resource wastage and increases the efficiency of the network operators. Similar nature of YouTube can be confirmed in [35]. In [34] it is evident that optimizing the traditional web and caching can improve the user experience.

In [24] and [5] authors did the study of video quality impact on user-behavior in large scale. In [24] authors showed how poor video quality influences user engagement, in what extent each KQIs influences the user engagement and how they vary across the video content type, genre and duration. They have collected the KQIs using client-side instrumentation and showed that total stalling ratio (buffering ratio) has the largest impact on user engagement across all content types but the degree of impact is more for live content. They have shown that users are ready to wait 3 minutes more for a 90-minute live video if there is a 1% reduction in buffering ratio. In [5] also authors showed that KQIs largely influence the user-behavior, especially on viewer abandonment rate and repeated viewship. They have shown that viewers start to abandon a video if it takes more than 2 seconds to start up and every one-second increase in the delay causes 5.8% users to abandon the video. A viewer plays 5% more video if we reduce 1% stalling (re-buffering) in the total video. Finally, they also showed that a viewer is 2.32% more likely to revisit the video service on the same site within a week if they did not experience a video failure. Some studies like [32] and [33] try to come up with the machine learning based model to predict the QoE. In [32], the authors analyzed the encrypted network traffic of YouTube and come up with an automated model to predict the QoE based on different network bandwidth parameters. They have shown that the 84% QoE classification accuracy with the extracted traffic features.

3.2 Motivation

To understand the user-behavior from network operators perspective, from [29] we have found that it is important to study how network parameters influence the user-behavior. And from [26] we have seen that studying only network events is not sufficient but we need to understand how playback events influence the user-behavior. From [24], [27] and [5] we understood that these studies have to be conducted on a large scale. For network operators to study the client side playback events in large scale it is hard to set-up the instrumentation as they do not usually have access to the client side player software and database. Many times these studies also expensive to conduct and may face some of the ethical and legal issues also if the network operators want to manipulate the certain conditions in the user network streams. To resolve these problems and to do a study on the impact of the user-
behavior on different network conditions which intern useful in predicting the video QoE, we are proposing an automated system which uses the existing user-behavioral models from large-scale user-behavioral studies. We have used the user-behavioral model from [5] and try to simulate it for different network conditions to study the impact on different key quality indicators (KQIs).

3.2.1 Overview of Large-Scale User-Behavioral QoE Monitoring

In conducting user behavioral studies in large scale there are many stack holders, instrumentation and infrastructures are required. In the studies by Krishnan et al. [5] also we can imagine such a set-up. To understand large-scale user-behavioral studies to measure the network impact in deeper we have come up with the flow diagram which would have been used by any network operators in conducting such a study.

The architectural diagram of such a set-up can be diagrammatically represented as in Figure 3.1. Here we can see each module needed to be connected globally by a big CDN network. Initially, there is a need to select a very diverse group of user/subjects from all over the world. After there is a need to tailor the network routes to broadcast video sample, video player software which collects the client side logs to these targeted groups. Once the network is installed and videos are sent to users, there is a need to collect logs to determine what user actions have carried out during the each video with corresponding network conditions. After collecting the data in server side and can come up with analytics which can be useful in determining the behavior of users corresponding to network conditions. This model can be useful later for better network service optimization.

Some of the limitations with this set-up is, it is very tedious, time-consuming and economically takes more effort to do so. While setting up the different network condition for different targeted groups, there may be a lot of technical, ethical and legal issues. Moreover, implementation of the client player software to do the monitoring and distribution of this software among different target groups is not only a challenging task but also requires the help of OTT content providers. To solve all these issues we are proposing a relatively simple and an automated monitoring system which can be used as an alternative in conducting large-scale user-behavioral studies for different network conditions.

3.2.2 Proposed Solution

In Figure 3.2 we can see the schematic diagram of the proposed solution in the thesis. It shows the implementation of automated set-up to conduct the user-behavioral video monitoring for different network conditions. Initially, we can simulate the selected user-behavior model using the automation frameworks such as Selenium or
Chrome extension and write a software to collect the video playback logs. Later we can feed the videos with different network conditions to this set-up and can collect the logs. Collected logs can be used to calculate the different KQIs and come up with the model to infer the impact of network conditions on these KQIs. This set-up can be repeated in the different geological location in case of location dependent network conditions. Inferred results can be used in optimizing the network service offerings which provides better user experience. This set-up has brought the earlier described set-up from globally distributed network to lab environment. It uses the available large-scale user-behavioral studies and applied that to prepare a user-behavioral model which can be later simulated in a lab environment.

3.3 Methodologies

In this thesis, we have followed the quantitative approach in conducting the research. Research methods and approaches are mainly experimental and deductive. We have collected the data by conducting the repetitive experiments for different network conditions. We have taken a combination of 21 different network conditions and used these values to calculate 8 different KQIs. Each experiment is repeated for 21 times to ensure the replicability of the collected data and analyzed using the statistical methods. In total, approximately 15000 videos are used in the analysis.
Figure 3.2: Proposed solution to measure the impact of network conditions contributing to 240 hours of content data.

3.3.1 User-Behavioral Model

The user-behavioral model is taken from the studies conducted by Krishnan et al. [5]. This study is conducted from the dataset collected from users all over the world using the Akamai’s client-side media plug-in. They have collected the playback events like startup delay (initial loading delay), re-buffering (stalling), play and pause states and user actions like closing the browser or tab. The data is from 6.7 million unique viewers including viewers across different connectivity like fiber, mobile, and DSL. They have used causality analysis over correlational analysis as they mentioned: "In fact, a purely correlational relationship could even lead one astray, if there is no convincing evidence of causality, leading to poor business decisions." To study the causal impact of quality on user-behavior they have designed a Quasi-Experimental Design (QED). While they have studied different user-behaviors, the one which we are interested in our thesis is 'Viewer Abandonment'.

Viewer abandonment indicates how long a user waits for the video to startup before he/she abandon watching the particular video. They have assumed the assertion "An increase in startup delay causes more abandonment of viewers" and found that the percent of abandoned viewers was positively correlated. The correlation plot is showed in Figure 3.3 A, which uses Kendall correlation with the correlation
value of 0.72.

The authors come up with a function to calculate the abandonment rate of the users for a given startup delay (initial loading delay) $x$. The Equation 1 gives the abandonment rate at $x$ initial loading delay. Refer [5] for more details. In Equation 1 Impatient($x$) is the views where users abandoned the video after less than $x$ seconds of initial loading delay and Patient($x$) is the views where users waited till $x$ seconds of initial loading delay.

$$\text{AbandonmentRate}(x) = 100 \times \frac{\text{Impatient}(x)}{\text{Impatient}(x) + \text{Patient}(x)} \quad (1)$$

In Figure 3.3 B authors used simple linear regression fitting to initial part of the curve where it shows all users waited 2 seconds before started abandoning the video and later every 1-second increase caused 5.8% of the users to abandon the video. After looking into the Kendall correlation plot in Figure 3.3 A, we can see that the data is more logarithmically distributed but the authors used simple first order linear regression which does not consider the impact of higher initial loading delay. Due to this reason, there was a formula developed by Werner Robitza [37] at Telekom Innovation Laboratories by tracing the data points given in the paper and used the inverse distribution to plot again. In Figure 3.4 we can see the plot which is plotted using inverse distribution and the abandonment rate is given by the Equation 2. From the Equation 2, we can say that users start abandoning the video at an initial loading delay of 1.8 seconds and 100% of the user would have abandoned if the initial loading delay becomes more than 64 seconds. Based on this equation we have modeled our simulated user-behavior.
Figure 3.4: Inverse regression fitting for abandonment rate v/s initial loading delay

\[
AbandonmentRate = 121.5 - \frac{1629}{InitialLoadingDelay + 11.59}
\]  

(2)

3.3.2 Different User-Behaviors

We have tried to model the user-behavior based on the model explained above and in this process, we have defined three different use cases in which the user might react in watching a video session. All our user behaviors are either assumed or simulated using the Chrome extension we have developed. We have not measured any real user-behavior. We have mainly divided these behaviors into three different flows namely Case 1, Case 2 and Case 3. These cases can be best explained below with the flow diagram.

1. Case 1: Here we have assumed that user wants to navigate to YouTube homepage and watch a single video for a viewing duration of 60 seconds and later decides to quit the browser. This is our passive single viewing session where we assumed user only wants to watch a single video per session without interacting with the browser. This can be represented using the flow diagram in Figure 3.5.

2. Case 2: In Case 1 we assumed that user watched only one video at a time and closed the browser. In a real-world scenario, it is not true that user always watches only one video at a time and reopens the browser again for watching a new video. In most of the scenario, user behavior is that they tend to watch many videos at one sitting. In the study [5] form which we have taken
the user-behavioral model, they have shown that the user on average watches 2.39 videos of an average duration of 22.48 minutes. In YouTube also one average mobile session lasts for more than 40 minutes [54]. But most of video QoE measurement always concentrate on measuring video QoE for only one video at a time and this may not depict the real-life scenarios and based on this we can’t make the decision on the user experience of a particular service. Hence we decided to measure multiple video viewing session at a time. In our experiment, we decided to measure watching 5 videos in a single viewing session as Case 2. Here we have assumed that user wants to navigate to YouTube homepage and start watching five consecutive videos by clicking on the next video after every 60 seconds. After watching 5 videos the user quits the browser. This is our passive multiple viewing session. The assumption here is user interacts with the browser only to go to the next video but not in case of initial loading delay or stalling (re-buffering). This can be represented using the flow diagram in Figure 3.6.

3. Case 3: In Case 2 to get the realistic measurement results we wanted to measure for multiple videos, but again this may not be an ideal user-behavioral measurement. The user also reacts to the situation and acts interactively. When a user sees the video is taking more time in loading he may react to this situation either by closing the web browser or trying to reload the web page to get the video to start to play again. Here to address the first scenario we are calculating how many percentages of users would be closed the browser.
by calculating the user abandonment rate. For the second scenario, we are simulating the user behavior of reloading the webpage and measure the new values. To do so, we have assumed that user navigates to YouTube homepage and start watching five consecutive videos by clicking on the next video after every 60 seconds. In the meantime, if the user finds a 5 seconds delay in starting up of any video, then they will choose to reload the web page and we are calling this action as "First reload". We have selected 5 seconds because according to Equation 2 5 seconds delay in the startup would have caused the 25% of the users to abandon the video session. We want to calculate the impact of user-behavior 'reloading the webpage' on initial loading delay of the reloaded page and how it relates to different network conditions. We will start measuring the new initial loading delay of the reloaded page and if the new value is less than 5 seconds we will finish that measurement and navigates to the next video. But if the new value is more than 5 seconds, then this time we will wait till initial loading delay becomes 10 seconds and if the initial loading delay exceeds 10 seconds then we will reload the web page again and we are calling this action as "Second reload". By 10 seconds of initial loading delay, according to the Equation 2 50% of the users would have left the session. This is the reason we want to calculate how many of these users can stay watching the session if they reload the webpage. The results can influence the behavior of the user and/or design of video players in case of worst initial loading delays.

**Figure 3.6:** Flow diagram for Case 2
After the second reload we start a new measurement and finish the 60 seconds. This can be represented using the flow diagram in Figure 3.7. From this case, we can calculate how many percentages of the user can be retained if we suggest/design the reload in the video watching session.
4 | Implementation

In this chapter, initially, we will give an overview of the modules used in the implementation of thesis and later we take each module and explain the tools required and software developed for designing it. After this, we have given an impression of the data collected during the experiments. Finally, we have discussed the hardware used for this thesis.

4.1 Architectural Diagram of the Implementations

In Figure 4.1 we can see the overview of the architectural design of this experimental set-up. Set-up can be divided into four different modules namely,

1. Network Module.
2. Server Module.
3. Client Module.
4. Data Analysis Module.

Each module is responsible for handling one particular task in the experimental set-up. Initially, a DSL network connection with maximum 50 Mbps download and 5 Mbps upload speed is taken. In order to simulate different network conditions in the setup (network module) a Linux tool called Wondershaper is used. Once the different network conditions are set in the client then the python script starts active probe script in ruby (server module) by passing the necessary parameters. Along with the active probe script, python script also launches a Chrome browser loaded with Chrome extension YouBot (client module) which generates automated user behavior. After this, a WebSocket connection is established between active probe script and YouBot. YouBot records the log events from the YouTube player and sends these logs back to active prove script using the WebSocket interface. In the server, these logs are used to calculate the different key quality indicators (KQIs) and are stored in the file system. Later data analysis module reads the data from the file system and does the data aggregation, plotting and automated model creation to create a formal relation between network conditions and user-behavior.

4.2 Network Module

In this experiment, our major goal is to calculate the video KQIs for different network conditions. In order to do this, we need to create the different network configurations consisting of the combination of upload and download speeds listed below.

<table>
<thead>
<tr>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload (Mbps)</td>
</tr>
<tr>
<td>Download (Mbps)</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>
Creating the network configurations for each different network conditions with the different network infrastructure is a challenging aspect and economically not feasible. This is one of the reasons for using network traffic control and shaping in a single server/client to emulate many different conditions. The initial step in this direction would be gaining a better control over the DSL connection. Network traffic shaping in the Linux in done at the kernel level and this can be achieved by using the Linux tool 'Traffic Control (tc)'. It manipulates the incoming traffic by controlling the network driver and output the controlled traffic over a required network interface. In Figure 4.2 we can see the block diagram explaining the basic stack of tc implementation. For more details we can refer [40], [41] and [42].

### 4.2.1 Wondershaper

In this experiment, we have used a tc based application for Linux called Wondershaper. Wondershaper [39] takes the network interface, upload, and download speeds
4 IMPLEMENTATION

Figure 4.2: Block diagram explaining tc stack [42]

![Block diagram](image)

**Figure 4.2:** Block diagram explaining tc stack [42]

Figure 4.3: Speed test using speedtest.t-online.de

![Speed test](image)

**Figure 4.3:** Speed test using speedtest.t-online.de

as input and shapes the traffic outgoing through that particular network interface. It also provides the commands to see and clear the traffic conditions applied to the particular network interface.

4.2.2 Speedtest

We have set up a DSL (Digital Subscriber Line) connection from the network provider and some of the statistics of the network setup can be seen here. The network connection is VDSL 50 with 50Mbps download and 5Mbps upload promised. Figure 4.3 shows the network statistics of connection measured by the speedtest.t-online.de [43] service offered by Deutsche Telekom. As seen in Figure 4.3 it shows the download speed of 46.88 Mbps, upload speed of 4.90 Mbps and ping of 18ms.
In Figure 4.4 we can see that comparison between current connection speed, technically possible bandwidth and maximum bandwidth in our tariff from the provider. It shows that connection can be varied in many scenarios and it is hard to assume service quality without knowing the network parameters. This is one of the reasons why we need to study user quality of experience with respect to the varying network parameters.

We understand that network quality is a varying parameter to measure and this is the reason we have repeated the measurement for 4 times and took average results. In Figure 4.5 we can see the average download speed of 46.805 Mbps, upload speed of 4.76 Mbps and a ping time of 18 ms.

To get the deeper understanding and to verify test results, we have done the ping test locally using ping command and speed test using python speed test library called Speedtest [44]. In Figure 4.6 we have pinged www.google.com and we have found the average ping of 34.8 ms.

To determine the download and upload speeds using Speedtest, we have executed commands in a python shell. The same library is used in our python script to determine and set the upload and download test cases of the experiment. In Figure 4.7 we can see that the reported download speed is 46.21 Mbps and upload speed of 3.79 Mbps.
4 IMPLEMENTATION

Figure 4.5: Diagram showing the average results

<table>
<thead>
<tr>
<th>Time</th>
<th>Download</th>
<th>Upload</th>
<th>Ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:35</td>
<td>46.83 Mbps</td>
<td>4.68 Mbps</td>
<td>18 ms</td>
</tr>
<tr>
<td>16:33</td>
<td>46.75 Mbps</td>
<td>5.02 Mbps</td>
<td>18 ms</td>
</tr>
<tr>
<td>16:30</td>
<td>48.88 Mbps</td>
<td>4.90 Mbps</td>
<td>18 ms</td>
</tr>
<tr>
<td>16:27</td>
<td>48.76 Mbps</td>
<td>4.44 Mbps</td>
<td>18 ms</td>
</tr>
</tbody>
</table>

Figure 4.6: Ping test using the Linux ping cmd

```bash
msqmsserver@msgms-test-server ~ $ ping google.com
PING google.com (172.217.17.238) 56(84) bytes of data.
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=1 ttl=56 time=35.2 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=2 ttl=56 time=34.9 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=3 ttl=56 time=35.0 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=4 ttl=56 time=34.9 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=5 ttl=56 time=34.4 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=6 ttl=56 time=34.7 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=7 ttl=56 time=34.8 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=8 ttl=56 time=34.9 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=9 ttl=56 time=34.6 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=10 ttl=56 time=34.7 ms
64 bytes from muc11s14-in-f14.1e100.net (172.217.17.238): icmp_seq=11 ttl=56 time=34.7 ms
^C
--- google.com ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10015ms
rtt min/avg/max/mdev = 34.419/34.848/35.221/0.259 ms
msqmsserver@msgms-test-server ~ $
```
4 IMPLEMENTATION

Figure 4.7: Speed test using the python lib Speedtest

4.2.3 Python Script

As seen in the Figure 4.1 we have written a python module which does speed test measurements and network tuning and shaping. Later it prepares the file storage and configuration file for the ruby server and launches Chrome browser with an open tab. While launching it also loads the two Chrome extensions namely YouBot which is developed in the project and uBlock [46] an efficient ad blocker extension for Chrome browser. It is used to block advertisements while playing the YouTube videos. As we were more concentrated on video continent QoE than advertisements, we have blocked them. One could do studies also with advertisements enabled in YouTube videos and how they also affect the user behavior and this study is out the scope of this project. Once measurements are done, it also kills both ruby server and Chrome browser and re-starts with new network configurations. This module repeats the experiment for each network configuration in a loop till it measures all different configurations of 21 independent variables for 21 times measuring a total of 15015 videos.

4.3 Server Module

The server module is based on the 'Active video probe' tool developed for Google Chrome by Werner Robitza [45]. It runs HTML5 and YouTube videos on the
browser. The script connects to the Chrome browser via a remote debugging protocol (using WebSockets) and logs the video playback events to the console, as indicated by the player API. Any HTML5 video can be played as long as the browser supports the codec. YouTube player uses the YouTube iFrame API, so it’s automatically compatible. The initial implementation was flexible enough to change and add some more playback events and extend the YouTube iFrame API based player to the actual YouTube player launched and handled by Chrome extension.

In the active probe script the main class `active_probe` is called from the python module and it creates a WebSocket connection to the Chrome browser using port 9200. Once it creates a successful connection it calls the class `probe_connection`. In `probe_connection` it prepares the JSON messages to send to Chrome over WebSocket in order to launch YouTube video player with a specific URL. This class also reads the log events form the WebSocket from Chrome and segregates them into network and console logs. In class `probe_results`, it calculates some of the KQIs using these log events and writes them to a JSON file on file system. In Figure 4.8 we can see the Unified Modeling Language (UML) diagram of the active probe and its interaction with the Chrome browser.

In the active probe, we have logged both console logs which give information about player events and network logs which gives information about network events form the browser. The following events are logged from probing script as a part of console logs in order to calculate KQIs:
4 IMPLEMENTATION

1. pageLoaded
2. documentReady
3. playerInitializing
4. playerReady
5. playerStateChange
6. playerUrlChanged
7. playerReloaded
8. playerQualityChange (from YouTube API)
9. videoDuration
10. finished

Using the above console events and the network events we can calculate different KQIs. To get an impression on calculated results from active probe please refer section 4.6.3.

4.4 Client Module

The major part of this project is to simulate the user-behavior similar to the actual users who are watching a video in order to do the QoE measurements to calculate the KQIs for different network conditions. In order to do this, we did an analysis of different user action flows in Section 3.3.2 and we tried to simulate these actions using Chrome events/clicks. The list of the user action that is possible during the video watching session are:

1. **Start the browser**: User decided to launch the browser in order to watch a video.

2. **Navigate to the YouTube URL**: User enters the YouTube URL in the browser and ends up in the homepage.

3. **Click on the video**: User may decide to click on the recommended video from the list in homepage.

4. **Start watching the video**: User starts watching the video for 60 seconds (we have considered this value in order to make the measurements faster but of course one can watch longer or till video ends).
5. **Click on the next video (passive watching):** User may decide to click on the next video in the list and watch the video (this may continue for five videos).

6. **Reload the current video (active watching):** User may decide to reload the video if he/she finds the loading of video taking longer. This may happen due to network changes.

7. **Close the browser:** User may decide to close the browser.

### 4.4.1 Generation of the Automated User-Behaviour

In order to automate the process, we tried to find the one to one mapping to the actions of the user and Chrome extension task. In Table 1 we can see that for every user action we have an automated counterpart to perform the same task. Later in the implementation, we will discuss in detail how we have generated these actions. In Figure 4.9 we can see the use case diagram explaining the all the three different cases.

<table>
<thead>
<tr>
<th>User Actions</th>
<th>Automated action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the browser</td>
<td>Python script launches the chrome browser</td>
</tr>
<tr>
<td>Navigate to the YouTube homepage</td>
<td>Ruby module sends the URL in JSON config through the WebSocket</td>
</tr>
<tr>
<td>Click on the video</td>
<td>YouBot module sends a click event on the web browser.</td>
</tr>
<tr>
<td>Start watching the video</td>
<td>YouBot keeps the time count (machine watching)</td>
</tr>
<tr>
<td>Click on the next video</td>
<td>YouBot detects the next video and clicks on it.</td>
</tr>
<tr>
<td>Reload the current video</td>
<td>YouBot measures the video initial loading delay relooads the player if the delay is 5 seconds initially and 10 seconds later.</td>
</tr>
<tr>
<td>Close the Browser</td>
<td>Python module kills the Chrome application.</td>
</tr>
</tbody>
</table>

**Table 1:** One to one mapping between user actions and automated actions.

### 4.4.2 Implementation of YouBot

YouBot is a Chrome extension to simulate the user-behaviors discussed in Section 3.3.2. Extensions are small software programs that can modify and enhance the functionality of the Chrome browser. We can write them using web technologies such as HTML, javascript, and CSS [47]. Extensions bundle its files into a single file that the user downloads and installs. This gives us the flexibility of running the extension
Figure 4.9: Architectural diagram of the user behavioral simulation
code independent of the web pages. They are essentially web pages, and they can use all the APIs that the browser provides to web pages, from XMLHttpRequest to JSON to HTML5. The extension can interact with web pages using asynchronous javascript and XML(AJAX) calls. We have used these features to create the actions we needed in the experiment. The general structure of the extension file has a manifest file and one or more optional javascript files. The manifest file called `manifest.json` gives information about the extension, such as the most important files and the capabilities that the extension might be using.

The extension includes the javascript files called content scripts which interact with web pages. In YouBot these content scripts read events from YouTube player using the YouTube iframe APIs and log the playback events. Some part of the content scripts also stores time duration and do the page reload or next video click actions after a particular time duration. In Figure 4.10 we can see that Chrome extension starts with the manifest file which loads all other files in the extension. `probe-init` javascript file loads the DOM and sends the page loaded event to the console and it calls the class file `active-prove` which is responsible for sending all events to console and later on WebSocket. It also initializes a player which reads events from YouTube iframe APIs and starts the YouTube service content script. YouTube content script is the one which is responsible for the generation of clicks, keeping track of event timings and storing and retrieving the data to and fro from session storage. Reload values are stored in the session storage because we also need those values when we reload web page in case of longer video startup times.

### 4.4.3 Youtube Iframe APIs

Iframe player API are the APIs developed by Google in order to provide the flexibility to embed any YouTube video inside another web page or player. We can control the player using javascript externally. Using API’s javascript functions, we can queue videos for playback, play, pause, or stop these videos and even adjust the player volume or retrieve information about the video being played [48]. We can also add event listeners that will execute in response to certain player events, such as a player state change or a video playback quality change.

Iframe APIs provides many javascript functions for playback control and player settings. It also possible to get the playback status, playback quality and information about player events using these functions. Some of the functions which are used in YouBot to get the playback events are explained below.

1. **Playback status**: `player.getPlayerState()` is a playback API which gives the state of the player at particular point of time. It returns the state of the player.

   Possible values correspond to the state of the player are listed in Table 2.
Figure 4.10: YouBot class diagram

<table>
<thead>
<tr>
<th>Return value</th>
<th>Player State</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>unstarted</td>
</tr>
<tr>
<td>0</td>
<td>ended</td>
</tr>
<tr>
<td>1</td>
<td>playing</td>
</tr>
<tr>
<td>2</td>
<td>paused</td>
</tr>
<tr>
<td>3</td>
<td>buffering</td>
</tr>
<tr>
<td>5</td>
<td>video cued</td>
</tr>
</tbody>
</table>

Table 2: Player states [48]
2. **Playback quality**: `player.getPlaybackQuality()` is a getter function which retrieves the actual video quality of the current video which is playing. Possible return values are `highres`, `hd1080`, `hd720`, `large`, `medium`, `small`, and `tiny`.

The API returns undefined if there is no current video or the video is not yet loaded. We can also set a playback quality using the setter API. Table 3 shows the playback quality levels that correspond to different standard player sizes.

<table>
<thead>
<tr>
<th>Return value</th>
<th>Player State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiny</td>
<td>Player height is 144px, and player dimensions are at least 144px by 256px for 16:9 aspect ratio.</td>
</tr>
<tr>
<td>small</td>
<td>Player height is 240px, and player dimensions are at least 320px by 240px for 4:3 aspect ratio.</td>
</tr>
<tr>
<td>medium</td>
<td>Player height is 360px, and player dimensions are 640px by 360px (for 16:9 aspect ratio) or 480px by 360px (for 4:3 aspect ratio).</td>
</tr>
<tr>
<td>large</td>
<td>Player height is 480px, and player dimensions are 853px by 480px (for 16:9 aspect ratio) or 640px by 480px (for 4:3 aspect ratio).</td>
</tr>
<tr>
<td>hd720</td>
<td>Player height is 720px, and player dimensions are 1280px by 720px (for 16:9 aspect ratio) or 960px by 720px (for 4:3 aspect ratio).</td>
</tr>
<tr>
<td>hd1080</td>
<td>Player height is 1080px, and player dimensions are 1920px by 1080px (for 16:9 aspect ratio) or 1440px by 1080px (for 4:3 aspect ratio).</td>
</tr>
<tr>
<td>highres</td>
<td>Player height is greater than 1080px, which means that the player’s aspect ratio is greater than 1920px by 1080px.</td>
</tr>
<tr>
<td>default</td>
<td>YouTube selects the appropriate playback quality.</td>
</tr>
</tbody>
</table>

Table 3: Player quality levels [48].

3. **Retrieving video information**: We have used two functions in retrieving
video information section. They are:

(a) `player.getDuration()` gives the duration in seconds of the current playing video.

(b) `player.getVideoUrl()` gives the URL for the current loaded/playing video.

4.5 Data Analysis Module

In the data analysis part, we read the data stored on file system from the ruby server including both server and network logs in order to calculate the video KQIs and in future these logs can also be used to calculate the MOS values. Data analysis module has three parts. In the initial part, we read all the data and format to make it easier to calculate the KQIs. In the second part, we have used python pandas framework do the analysis and plotting. pandas is an open source library providing high-performance, easy-to-use data structures, and data analysis tools for Python programming language [49]. At last, we have used non-linear curve fitting to build the automated learning model to predict the different KQIs for different network conditions.

The list of KPIs calculated in the analysis part is:

1. Initial Loading Delay
2. Average Stalling Duration
3. Total Stalling Duration
4. Stalling Events
5. Number of Stalling Events
6. Quality Events
7. Number of Quality Events
8. User Abandonment Rate
9. Total Stalling Ratio (Buffering ratio)
10. Stalling Ratio (Re-buffering ratio)
11. Playout Duration of Each Quality
12. Total Playout Duration of video
13. Page Reload Events
14. Average Audio Bitrate
15. Average Video Bitrate

4.5.1 Data Selection and Aggregation

We had collected 45,000 data files. Initially, we have loaded the data into a pandas dataframe and did the data selection. As we have repeated the each experiment with the different conditions for 21 times, the data section part was necessary to remove outliers. We have used box plots to help in removing these outliers. We have plotted vertical box plots as we wanted to remove outliers for each KQIs along the y-axis for a particular download/upload combination along the x-axis. In box plots, the bottom and top of the box are always first and third quartiles, and the band inside the box is always the second quartile (the median). In our plots, we have used the 1st percentile and 99th percentile for box quartiles. For data aggregation part we have always calculated the mean of points with the confidence interval of 97.5%.

4.5.2 Plots

To plot all the graphs we have used python plotting libraries matplotlib and pyplot. We have plotted box, bar and line plots to visualize all data points and use them in analysis part. Initially, all KQIs are plotted against each upload or download speeds by keeping other variables constant. All the three use cases are plotted separately and later we have done a comparison of these plots to get a better understanding.

4.5.3 Automated Learning Model

To create a prediction model for video QoE and different network conditions, initially we have tried simple linear regression [51] and least median square regression [52] machine learning algorithm and the results obtained with these were not very convincing as the correlation values we got were very lower. This is may be due to the non-linear nature of the data and this can also be seen in the KQI plots. Later we decided to use non-linear curve fitting [50] as the tool for creating the prediction model. We found out that these models have the good correlation values. To do the comparison between the results we have used Pearson correlation coefficient and RMSE values for different KQIs are tabulated in Section 5.4.

4.6 Impression of the Data Collected

In our experiment, we have network and console logs. Console logs consist of playback events from YouTube player. Network logs consist of network events collected
4 IMPLEMENTATION

Figure 4.11: Screen shot showing the video playing in the YouTube from the Chrome browser.

4.6.1 Console logs

In Figure 4.11 we can see the screen-shot of the measurement process showing the console logs which has the collected playback event details.

Console logs from one of the experiment:

Connected to WebSocket: ws://127.0.0.1:9223/devtools/page/e7128a9b–bcf2–4b1b–9ee8–1d6195eb387f
Sent message: {method=>'Network.enable', 'id'=>0}
Sent message: {method=>'Console.enable', 'id'=>1}
Sent message: {method=>'Runtime.enable', 'id'=>2}
Sent message: {method=>'Page.enable', 'id'=>3}
Sent message: {method=>'Page.navigate', :params=>{:url=>'https://www.youtube.com'}, 'id'=='4'}
Received message – console: {"event":"documentReady","message":"DocumentReady","data":","timestamp":1496236221259}
Received message – console: {"event":"pageLoaded","message":"Pageloaded","data":",timestamp":1496236244024}
Received message – console: {"event":"domQueryServed","message":"The DOM query video has
4 IMPLEMENTATION

been served after 27022ms,"data":{"query":"video","time":27022,"time_unit":"ms"},"timestamp":1496236271062,"source":"DOMServer"}

Received message – console: {"event":"playerInitializing","message":"Player is initializing","data":","timestamp":1496236271064,"source":"html5"}

Received message – console: {"event":"playerUrlChanged","message":"Player url changed to https://www.youtube.com/watch?v=u0kYDmHid3s","data":https://www.youtube.com/watch?v=u0kYDmHid3s","timestamp":1496236244039,"source":"youtube"}

Received message – console: {"event":"playerReady","message":"Player ready","data":","timestamp":1496236271049,"source":"youtube"}

Received message – console: {"event":"playerQualityChange","message":"Player quality changed to tiny","data":"tiny","timestamp":1496236271052,"source":"youtube"}

Received message – console: {"event":"playerStateChange","message":"Player state changed to stalling","data":"stalling","source":"html5","timestamp":1496236271065}

Received message – console: {"event":"videoDuration","message":"Video duration is 2881.666031746032","data":2881.666031746032,"timestamp":1496236306597,"source":"youtube"}

Received message – console: {"event":"playerStateChange","message":"Player state changed to playing","data":"playing","source":"youtube","timestamp":1496236306598}

Received message – console: {"event":"playerStateChange","message":"Player state changed to stalling","data":"stalling","source":"youtube","timestamp":1496236316620}

Received message – console: {"event":"playerStateChange","message":"Player state changed to playing","data":"playing","source":"youtube","timestamp":1496236321815}

Received message – console: {"event":"playerStateChange","message":"Player state changed to stalling","data":"stalling","source":"youtube","timestamp":1496236324102}

Received message – console: {"event":"finished","message":"Probing finished","data":","timestamp":1496236331067,"source":"html5"}

Probing finished. Calculating results.

4.6.2 Network logs

Along with the console logs, we also logged the network events from the Chrome browser. In Figure 4.12 we can see the screen-shot of the measurement process showing the network events in the browser.

Sample of the network log from one of the experiment. It only shows the two events from the complete network logs:

```
{"timestamp":785.127784,"remoteIPAddress":"173.194.151.88","content_type":null,"url":"https://r2...0","itag":\["278"\],"mime":\["video/webm"\],"range":\["0-65972"\],"dur":\["62.240"\]}
{"timestamp":817.913099,"remoteIPAddress":"173.194.151.88","content_type":null,"url":"https://r2...0","itag":\["278"\],"mime":\["video/webm"\],"range":\["0-65972"\],"dur":\["62.240"\]}
```

4.6.3 Output of Calculated KQIs

Active probe script do some of the KQI calculation and saves to the file system in JSON format. Sample JSON file:
Figure 4.12: Screen shot showing the network events in the browser

```json
{
"player_ready_timestamp":1497544973037,
"player_load_time":2,
"startup_delay":10935,
"video_duration":2357,
"video_url":"https://www.youtube.com/watch?v=OxginLSLGOE",
"average_stalling_duration":10290.5,
"stalling_events":[
    [1497544973039, 10935],
    [149754991121, 9646]
],
"quality_events":[
    [1497544973063, 'tiny']
}
```
4.7 Hardware Used

Here is the technical specification of the hardware used in this thesis. The whole experimental setup is built on a single embedded fanless system from lex systems[7]. The TINO 2PCI + CI170C systems are compact and power saving systems which can be easily installed in many different locations for the video QoE monitoring. These systems operate with the +12V DC at a temperature ranging from -20 to 60 degree Celsius. The technical specification of the system are listed in the Table 4.

<table>
<thead>
<tr>
<th>System info</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Linux Mint 17.3 Cinnamon 64-bit</td>
</tr>
<tr>
<td>Cinnamon Version</td>
<td>2.8.8</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>3.19.0-32-generic</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Core i7-3612QM CPU @ 2.10Ghz x 4</td>
</tr>
<tr>
<td>Memory</td>
<td>DDR4 2133MT/s, 8GB</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>120GB SSD</td>
</tr>
<tr>
<td>Networking</td>
<td>2xGb Ethernet</td>
</tr>
</tbody>
</table>

Table 4: Technical specification of the embedded hardware used.
5 | Results and Analysis

In this section, we present a detailed analysis of obtained results from measurements described in the previous chapters. We are making a details comparison of the results obtained from different cases of user-behaviors explained in Section 3.3.2. Later we will present the results and comparison of learning model explained in the Section 4.5.3.

We are initially going to discuss the impact of different network conditions on KQIs. Here we are going to include the results from Case 1 explained in Section 3.3.2. In the later sections, we are going to make a comparative analysis of each case on different KQIs.

5.1 Impact of Network Conditions on KQIs

5.1.1 Initial Loading Delay

As defined in Section 2.9 initial loading delay is time taken for the video to start after the user clicked on the video play button. Impact of varying upload and download speeds on initial loading delay can be explained below.

In Figure A.1 we can see box plots for different download speeds by keeping upload speed constant. In these figures, we can see that lower upload speed gives more variability in initial loading delay both lower and higher download speeds.

In Figure 5.1 we can see changes in initial loading delay (startup delay) for upload speed by changing download speeds across the x-axis. Form the figure we can infer that increasing the download speed varies the initial loading delay non-linearly with a negative slope. This behavior is consistent with all the different upload speeds. One noticeable behavior in the curve is how the change in the upload speed impact initial loading delay in higher download speeds. Having low upload speed has a strong impact on initial loading delay even in higher download speeds like 30 Mbps. It is even evident that initial loading delay for low upload speed increases linearly with increase in download speed after a certain point (in Figure 5.1 after 8 Mbps).

5.1.2 Average Stalling Duration

As defined in Section 2.9 average stalling duration is the average of all stalling including initial loading delay in the video session. Impact of varying upload and download speed on average stalling duration can be explained below.

The box plots for different download speeds by keeping upload speed constant followed the same trend as initial loading delay. In this case, also we can see that smaller upload speed gives more variability in average stalling duration even in larger download speeds. In Figure A.2 we can see box plots for different upload speeds by
5 RESULTS AND ANALYSIS

Figure 5.1: Impact of upload/download speed on initial loading (startup) delay keeping the download speed constant. In these figures we can see that mid-range download speeds i.e 0.3 Mbps to 2 Mbps has more variations irrespective of upload speed.

In Figure 5.2 we can see changes in average stalling duration for upload speed by changing download speeds across the x-axis. Here we can notice that lower upload speed has a strong impact even in higher download speeds like 30 Mbps. This can also be explained similarly to initial loading delay.

5.1.3 Total Stalling Duration

As defined in 2.9 total stalling duration is the sum of all stalling including initial loading delay of the video session. Impact of varying upload and download speed on total stalling duration can be explained below.

The box plots of total stalling duration also followed the same trend as initial loading delay and average stalling duration. In Figure 5.3 we can see changes in the upload speed for all the different combinations of download speeds. Here also we can see that lower upload speed has a strong impact even in higher download speeds.
5 RESULTS AND ANALYSIS

Figure 5.2: Impact of upload/download speed on average stalling duration

Figure 5.3: Impact of upload/download speed on total stalling duration
5 RESULTS AND ANALYSIS

5.1.4 Number of Stalling
As defined in Section 2.9 number of stalling is the total number of stalling that occurred after the video started playing. Impact of varying upload and download speed on the number of stalling can be explained below.

As seen in Section 5.4, we can see that lower upload speeds have less number of stalling once the video starts playing even in lower and higher download speeds. In Figure 5.4 we can see that red curve i.e for upload speed of 0.256 Mbps has very less impact on the number of stalling compared to high upload speed dark green curve of 3.3 Mbps. This behavior we can later explain with the help of quality switches. One conclusion we can get from this graph is that during lower upload speed videos takes a longer time to load but once it gets loaded, there high possibility that they are going to play without stalling again in between. This may be due to the YouTube player behavior discussed in [31] that in mid range or higher. YouTube player starts sending more number of a small video stream of different quality to check the network speed during this time there is a high traffic and more network wastage [31] happens. We are guessing YouTube starts adapting to the network speed only when it finds a minimum upload and download speed or mid-range i.e from 0.5 Mbps. For all lower speeds, it just plays the lower quality and does not try to adapt in middle to increase the quality level of video. This may be the reason in mid-range download speeds (0.3 Mbps to 2 Mbps) there is a lot of probability that video may face again with 1 or more stalling in between due to quality switching.

5.1.5 Number of Quality Switches
As defined in Section 2.9 number of quality switches is the total number of video quality level changes occurred during the video session. Impact of varying upload and download speed on the number of quality switches can be explained below.

As seen in Figure 5.5 for lower download speeds we have fewer quality switches and in mid-range of download speeds there are more quality switches as high as 3 to 4. We can also notice that there is again a spike in the quality switches in 2 Mbps to 4 Mbps. The behavior of upload speed variation is consistent with previous KQIs as we can see more number of quality switches for lower upload speed even during the higher download speed. The notable behavior to observe is quality switches for lower upload and higher download speed combination is still lower than all upload and mid-range download speeds. This explains variations in the number of stalling in Figure 5.4. As explained in the Section 5.1.4 the player is trying to adapt to the changes in the network speeds in the mid-download ranges from 0.5 to 3 Mbps. This adaptive behavior causes the more stalling which occurred in Figure 5.4.

To understand in depth about quality switching behavior of different network conditions, we have analyzed each quality playout duration with respect to total
5 RESULTS AND ANALYSIS

Figure 5.4: Impact of upload/download speed on number of stalling

Figure 5.5: Impact of upload/download speed on number of quality switches
video playout duration i.e if total video viewing duration was 60 seconds then how much time each quality video got played.

Below we are going to see viewing duration of different quality levels as listed in Table 3. There are 9 different quality levels that can be possible in YouTube videos. They are default, highres, hd1080, hd720, large, medium, small and tiny, but default, highres got never played during our experiment.

In Figure A.3 we can see bar plots for different download speeds by keeping the upload speed constant. Here we can see more than two types of quality resolution videos are getting played in mid-range of download speed and this behavior is intact throughout different upload speeds.

In Figure A.4 we can see bar plots for upload speeds by keeping download speed constant. Here we can see that during lower download speed even duration of total video played is less and there are only few quality switches. The overall video played duration is different for different download speeds due to higher startup and stalling duration. In mid-range download speeds there are many quality switches and again for higher download speed number of quality switches reduces.

5.1.6 Total Stalling Ratio

As defined in Section 2.9 total stalling (buffering ratio) ratio is the ratio of time spent in video buffering to total video duration. Impact of varying upload and download speed on total stalling ratio can be explained below.

In Figure 5.6 we can see changes in upload speed for all different combinations of download speeds on the same graph. Here we can notice that low upload speeds have a high impact on the total stalling ratio. In mid-range download speed, there are noticeable fluctuations in total stalling ratio.

5.1.7 Stalling Ratio

As defined in Section 2.9 stalling (re-buffering) ratio is the ratio of time spent on video buffering after the initial loading delay to total video duration. Impact of varying the upload and download speed on stalling ratio can be explained below.

As mentioned in [5] stalling (re-buffering) ratio plays an important role in user-behavior and user-retainment ratio for a particular OTT service. In Figure 5.7 we can see that lower upload speeds have lower stalling ratio even both in lower and higher download speeds. In Figure 5.7 we can see that red curve i.e for upload speed of 0.256 Mbps has little impact on the stalling ratio compared to high upload speed dark green curve of 3.3 Mbps. This behavior can be explained with more number of stalling in these conditions. Stalling ratio has no impact on high download speeds. High upload speed with lower or mid-range download speeds has a high impact on
5 RESULTS AND ANALYSIS

5.1.8 User Abandonment Rate

As defined in Section 2.9 user abandonment rate is the percentage of viewer who starts to stop watching the video after a certain amount of initial loading delay. The user abandonment rate has a positive correlation with the initial loading delay [5]. User Abandonment Rate is given by Equation 2. This is the user-behavior model which we explained in Section 3.3.1

As seen in Figure 5.8, we can see that lower upload speeds have around 80% of the User Abandonment Rate that means for these values there is an 80% chance that user stops watching the video before even it loads. This behavior has a negative linear effect as we go forward to higher upload and download speeds. We can also see for lower upload speeds even in higher download speeds there is a 15% chance that user is going to stop the video. This shows that even if we have higher download speed, the upload speed has a high impact if it is low and even it increases inversely after certain download speed (in the Figure 5.8 at 8 Mbps).

**Figure 5.6:** Impact of upload/download speed on total stalling(buffering) ratio

stalling and this may be due to the YouTube player behavior of trying to adapt to the changing network conditions.
5 RESULTS AND ANALYSIS

Figure 5.7: Impact of upload/download speed on stalling (re-buffering) ratio

Figure 5.8: Impact of upload/download speed on user abandonment rate
5 RESULTS AND ANALYSIS

5.2 Impact of Measuring Single video session compared to the Multiple video session on KQIs

As discussed in Section 3.3.2 users always do not watch only one video in one session they try to watch multiple videos in one session [5] [54]. In our experiment, we decided to measure watching 5 videos in a single viewing session as Case 2. Here we are going present detailed comparison results from Case 1 and Case 2.

5.2.1 Initial Loading Delay

In Figure 5.9 we can see that initial loading delay for lower download speeds for all different upload speeds in Case 2 is around 35% less compared to same download speeds in Case 1. This number has an exponential decrease till higher download speeds. This shows that when watching multiple videos even during slower network connections, videos start quickly. This may be a YouTube specific behavior and maybe also depends on their CDN architecture. This behavior can be explained with the help of [34] where the author has shown that using the advanced cache options can reduce YouTube’s core server infrastructure. Applying such a caching mechanism would have affected this behavior and in [20] there is a discussion of YouTube bitrate adaptation algorithm which also may have an impact on how YouTube player would have already adopted to network conditions. In case of multiple video session, after the first video there is no need for the player to initially download the initial burst [21] which helps in throttle downing the bitrate. Player adaptation also causes the increase of redundant traffic in the network [31], increase in the redundant traffic also slows down the network. This behavior may also be dependent on the caching selection policies employed by Google’s CDN, according to [23] this also has an effect on video QoE. This result needs to be further studied in order to explain how it is different for multiple video sessions, but this study shows how it is important to conduct the QoE measurement not only on a single video but multiple videos at a time. One more interesting result from Figure 5.9 is that for high upload and download speeds, initial loading delay seems to be more for the Case 2 compared to Case 1. This is a very interesting result in the direction of determining how multiple video session affects the user-behavior in case of higher upload and download speeds.

5.2.2 Average Stalling Duration

Average stalling duration follows same trend as initial loading delay. It can be seen in Figure 5.10.
5 RESULTS AND ANALYSIS

Figure 5.9: Comparison of case 1 and 2 on initial loading delay

Figure 5.10: Comparison of case 1 and 2 on average stalling duration
5.2.3 Total Stalling Duration

Total stalling duration also follows the same trend as initial loading delay. It can be seen in Figure 5.11.

5.2.4 Number of Stalling

In Figure 5.12, we can see that number of stalling remain same for lower download speed but in mid-range download speed there was more randomness in the number of stalling in Case 1 in Figure 5.4 and this has been reduced and become nearly linear across this range. There is an around 30% reduction in the number of stalling for Case 2 for mid-range download speeds. As discussed in Section 5.2.1 we can relate this to the YouTube player behavior as it already knows the network bandwidth, it may be not trying again to adapt to the network conditions. One more observation in Figure 5.12 is the number of stalling got increased for higher upload and download speed compared to the Case 1.

5.2.5 Number of Quality Switches

In Figure 5.13, we can see that number of quality switches reduced for Case 2 for all upload and download speeds, but it is hard to notice in high download and low upload case. One of the interesting behaviors of the curve is that for all upload and
download combinations of the curve for Case 1 and Case 2 follows the same trend with an around 20% less number of quality switches. As we discussed in the Section 5.2.1 we assumed that the YouTube player is caching the network conditions and it is not trying to adapt to the new quality levels and this can also be get confirmed with the reduced number of quality switches in case of Case 2. To do more detailed analysis we also plotted the time duration of each quality played with respect to total video duration.

In Figure A.5 we can see bar plots for different download speeds by keeping the upload speed constant. In the figure, we can see time duration at each quality for lowest and highest upload speeds. Here we can notice that total viewing duration for low upload and download speed is more in Case 2 compared to Case 1, but for higher upload and download speed we can see more viewing duration for Case 1 compared to Case 2. In Figure A.6 we can see that Case 2 have less number of quality switches compared to Case 1 for low download speeds. In mid-range download speeds, we have more quality switches but the once selected quality played longer in Case 2.

5.2.6 Total Stalling Ratio

In Figure 5.14 we can see that total stalling (buffering) ratio also follows the same trend as initial loading delay discussed in Section 5.2.1 with around 30 to 35% reduction for lower upload and download speeds and we also can see increase in
5 RESULTS AND ANALYSIS

Figure 5.13: Comparison of case 1 and case 2 on number of quality switches

total stalling ratio for case 2 for higher download speeds.

5.2.7 Stalling Ratio

In Figure 5.15 stalling (re-buffering) ratio curve for Case 1 and Case 2. There is no reduction in the stalling ratio for lower upload and download speeds this means that even for multiple video session once the video loaded there is a chance that it will play without interruptions. But stalling has been reduced around 40% for mid-range download speeds and again there is an increase in stalling for higher download speeds due to increase in the number of stalling.

5.2.8 User Abandonment Rate

In Figure 5.16 we can see that user abandonment rate for both Case 1 and Case 2. In the figure, we can see that for Case 2 it has been reduced by 10% for download speeds below 8 Mbps. This means that the users who are watching multiple videos are more tolerant to the changes in the network fluctuations compared to the one watching a single video. After 8 Mbps the user abandonment ratio has become zero. This means the users who are better connected tend to stay longer watching the video.
Figure 5.14: Comparison of case 1 and 2 on total stalling (buffering) ratio

Figure 5.15: Comparison of case 1 and 2 on stalling (re-buffering) Ratio
5.3 Impact of Reloading on KQIs

In Section 3.3.2 we have a Case 3 where we assumed that user reacts to the situation and acts interactively. When a user sees startup delay more than 5 seconds he tries to reload the video. After reload he continue to watch and again if he faces an initial loading delay of more than 5 seconds, this time he is going to wait till 10 seconds and tries to reload the video and continue watching. In this section, we are going to see the result of comparison between Case 2 and Case 3.

5.3.1 Initial Loading Delay

In Figure 5.17 we can see that initial loading delay got increased a bit for Case 3 compared to Case 2 in lower upload and download speeds. For mid-range download speeds, it got reduced around 20% especially for the download speed spanning from 0.5 to 2 Mbps. There is a download speed range where we got a higher number of stalling and quality switches in both Case 1 and Case 2. As we discussed in Section 5.2.1 during this range of download speeds the player is trying to adapt to the changing network speeds and may be trying to play a higher quality video. When we reload the web page, by this time it already has the network conditions and it may adapt to known quality and this is just an assumption and it needs to be studied in detail. This shows that if users tend to refresh web page during mid-range downloads, they are going to see an improvement in their quality of experience. In
5 RESULTS AND ANALYSIS

Figure 5.17: Comparison of case 2 and 3 on initial loading delay

Figure also we can see that lower upload and higher download speeds part of the curve have lower initial loading delays compared to Case 2.

5.3.2 Average Stalling Duration

Average stalling duration follows same trend as initial loading delay in Case 2 and 3 also. It can be seen in Figure 5.18.

5.3.3 Total Stalling Duration

Total Stalling Duration also follows the same trend as initial loading delay. It can be seen in Figure 5.19.

5.3.4 Number of Stalling

In Figure 5.12, we can see that number of stalling got increased in lower download speed and got reduced in higher download speeds compared to Case 2. One more interesting observation we noticed is in Case 2 where the number of stalling got increased in higher download speeds are got reduced after reloading the page. From this, we can conclude that users going to see more stalling if he tries to reload the player in lower download speed and fewer stalling in higher download speeds.
5 RESULTS AND ANALYSIS

Figure 5.18: Comparison of case 2 and 3 on average stalling duration

Figure 5.19: Comparison of case 2 and 3 on total stalling duration
5 RESULTS AND ANALYSIS

5.3.5 Number of Quality Switches

In Figure 5.21, we can see that number of quality switches reduced for Case 3 for mid-range download speeds. However, this analysis can not be completed without looking at the video duration of each quality levels.

In Figure A.5 we can see bar plots for different download speeds by keeping the upload speed constant. In the figure, we can see time duration at each quality for midrange download and highest upload speeds. Here we can notice that total viewing duration varies i.e more number of quality switches in mid-range of 1.5 to 2 Mbps. In Figure A.6 we can see that Case 3 have less number of quality switches compared to Case 2 for low download speeds. In mid-range download speeds, we have more quality switches but the once selected quality played longer in Case 3.

5.3.6 Total Stalling Ratio

In Figure 5.22 we can see that there is a slight reduction in total stalling (buffering) ratio of around 5% for mid-range download speeds and higher download and upload speeds.
5 RESULTS AND ANALYSIS

Figure 5.21: Comparison of case 2 and 3 on number of quality switches

Figure 5.22: Comparison of case 2 and 3 on total stalling (buffering) ratio
5 RESULTS AND ANALYSIS

5.3.7 Stalling Ratio

In Figure 5.23 shows that reloading video player helps in reducing stalling (re-buffering) ratio in the higher download and upload speeds.

5.3.8 User Abandonment Rate

In Figure 5.24 we can see that there is a 5 to 8% reduction in user abandonment rate for Case 3. Even we can notice that reloading helps for the users of higher download and upload speeds. In Case 3 we can say user abandonment become 0% after download speed of 10 Mbps and upload of 1 Mbps.

5.3.9 Impact of Reload

In Figure 5.25 we can see graph showing percentage of videos never reached reloading stage i.e they have initial loading delay less than 5 seconds. In lower download and upload speeds all videos got reloaded and this got gradually increased towards the higher upload and download speed.

In Figure 5.26 we can see the percentage of video which loaded within 5 seconds after first reload. This is the indication of getting a better user experience for the user as the video loaded after reloading the web page. We can notice that during mid-range download speeds around 40 to 60% of videos gave better performance.
RESULTS AND ANALYSIS

Figure 5.24: Comparison of case 2 and 3 on user abandonment rate

Figure 5.25: Percentage of videos loaded after initial loading delay < 5s
with respect to the total number of videos got reloaded. In Figure 5.27 we can see percentage of video which gave better experience for the user after the second reload compared to total number of video which got reloaded. In this case user reloaded the page two times in total and got the video loaded. Here we see 100% in range later 1.5 Mbps that is because number of videos got reloaded also got drastically reduced after first reload in case of high download speeds.

Figure 5.28 shows percentage of videos which never got better even after second reload. Here we can see that 80% of video in low upload and download speeds get the worse state after both the reloads. Reloading only helps either in mid-download speeds or high download speeds. This can be seen in the increased initial startup delay case Case 3 compared to Case 2. Also, these values only reflect the lower upload and download speeds and we only tested these values by varying the network conditions and these values do not include any simulated CDN problems or any YouTube server failures which can also cause video loading and failure problems for the users. CDN related study is done in [23] and these studies can also apply with respect to the user-behavioral changes.
5 RESULTS AND ANALYSIS

Figure 5.27: Percentage of videos loaded after initial loading delay < 10 s after second reload

Figure 5.28: Percentage of videos loaded after initial loading delay > 10 s even after second reload
5.4 Models to predict the KQIs

We have created the automated models to predict video KQIs based on the network parameters. We used the Non-Linear Least-Squares Minimization and Curve-Fitting for Python library LMFIT [55]. It provides the non-linear optimization and curve fitting using the Levenberg-Marquardt method [55]. It is used to predict different KQIs using network parameters and we have ignored the effect of upload speeds as they only affect the high download speeds. Due to the nonlinear nature of the data, we have used the triple exponential equation and line equation in the LMFIT composite models. Composite models can be obtained by adding or combining the two or more models using the algebraic operations and parameters from individual models influence the whole model [55].

The composite model is given by Equation 3

\[
[\text{Model}] = \text{slope} \times (a \times \exp(\exp(b \times x + c))) + d + \text{intercept}
\]  

(3)

where \( x \) = download speed in Mbps

Equation 3 is used for all the KQIs for all three cases with different initial conditions.

5.4.1 Initial Loading Delay

In Equation 3 we can see the curve fitting equation for all three Cases for initial loading delay for different initial conditions. In these equations we have ignored the effect of upload speeds as they only affect the high download speeds. In Table 5 we see the list of coefficients and initial conditions for the Equation 3. Table 6 will give the correlation values obtained for the data and in Figure 5.29 we can see the curve fitting graph for different cases.

<table>
<thead>
<tr>
<th>Cases</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>slope</th>
<th>intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.9952</td>
<td>-0.85881</td>
<td>0.4042</td>
<td>3.8179</td>
<td>-0.1214</td>
<td>-1.9021</td>
</tr>
<tr>
<td></td>
<td>(init= 1.2)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.5441</td>
<td>-1.1117</td>
<td>0.4797</td>
<td>1.3893</td>
<td>-0.0543</td>
<td>0.48174</td>
</tr>
<tr>
<td></td>
<td>(init= 0.8)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.4389</td>
<td>-1.2239</td>
<td>0.5818</td>
<td>1.9277</td>
<td>-0.0543</td>
<td>-0.1592</td>
</tr>
<tr>
<td></td>
<td>(init= 0.8)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
</tbody>
</table>

Table 5: Table of coefficients for initial loading delay

Initial conditions, fit statistics and variables and their correlation for individual cases for initial loading delay can be given below:

Case: 1
5 RESULTS AND ANALYSIS

Fit Statistics:

# function evals = 207
# data points = 52
# variables = 6
chi-square = 107.578
reduced chi-square = 2.339
Akaike info crit = 49.802
Bayesian info crit = 61.510

Variables:

a: 0.99526034 +/- 0 (0.00%) (init = 1.2)
b: -0.85881312 +/- 0 (0.00%) (init = -0.3)
c: 0.40421920 +/- 0 (0.00%) (init = 0.3)
d: 3.81791467 +/- 0 (0.00%) (init = 0.91)
slope: -0.12148530 +/- 0 (0.00%) (init = 0.05)
intercept: -1.90216363 +/- 0 (0.00%) (init = 0)

Case: 2

Fit Statistics:

# function evals = 193
# data points = 52
# variables = 6
chi-square = 25.319
reduced chi-square = 0.550
Akaike info crit = -25.424
Bayesian info crit = -13.717

Variables:

a: 0.54419463 +/- 2.983994 (548.33%) (init = 0.8)
b: -1.11171465 +/- 2.571717 (231.33%) (init = -0.3)
c: 0.47970298 +/- 0.559481 (116.63%) (init = 0.3)
d: 1.38932471 +/- 0.013096 (24.08%) (init = 0.91)
slope: 0.48174261 +/- 2.13e+06 (443106334.42%) (init = 0)
intercept: 0.48174261 +/- 2.13e+06 (443106334.42%) (init = 0)

Correlations: (unreported correlations are < 0.100)

C(d, intercept) = -1.000
C(a, b) = -1.000
C(a, c) = -0.999
C(b, c) = 0.998
C(b, slope) = 0.632
C(a, slope) = -0.621
C(c, slope) = 0.610
C(b, intercept) = 0.434
C(b, d) = -0.434
C(a, intercept) = -0.432
C(a, d) = 0.432
C(c, intercept) = 0.428
C(c, d) = -0.428
C(slope, intercept) = 0.284
C(d, slope) = -0.284
Case: 3

Fit Statistics:

- # function evals = 255
- # data points = 52
- # variables = 6
- $\chi^2$ = 16.451
- reduced $\chi^2$ = 0.358
- Akaike info crit = $-47.845$
- Bayesian info crit = $-36.138$

Variables:

- a: $0.43897775 +/- 2.333329$ (531.54%) (init= 0.8)
- b: $-1.22397516 +/- 2.685633$ (219.42%) (init= -0.3)
- c: $0.58183412 +/- 0.337640$ (58.03%) (init= 0.3)
- d: $1.92773010 +/- 1.12e+06$ (58189161.37%) (init= 0.91)
- slope: $-0.05437673 +/- 0.011525$ (21.20%) (init= 0.05)
- intercept: $-0.15929753 +/- 1.12e+06$ (704175682.94%) (init= 0)

Correlations: (unreported correlations are < 0.100)

- C(d, intercept) = -1.000
- C(a, b) = -1.000
- C(a, c) = -0.998
- C(b, c) = 0.997
- C(b, intercept) = 0.751
- C(b, d) = -0.751
- C(a, intercept) = -0.742
- C(a, d) = 0.742
- C(c, intercept) = 0.720
- C(c, d) = -0.720
- C(slope, intercept) = 0.679
- C(d, slope) = -0.679
- C(b, slope) = 0.667
- C(a, slope) = -0.657
- C(c, slope) = 0.635

<table>
<thead>
<tr>
<th>Cases</th>
<th>PCC</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.9857</td>
<td>1.4383</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.9886</td>
<td>0.6977</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.9940</td>
<td>0.5624</td>
</tr>
</tbody>
</table>

Table 6: Results for initial loading delay

5.4.2 Average Stalling Duration

In Table 7 we see the list of coefficients and initial conditions for the Equation 3. Table 8 will give the correlation values obtained for the data. In Figure 5.30 we can see the curve fitting graph for the given equation for average stalling duration.
Figure 5.29: Curve fitting for initial loading (startup) delay
## 5 RESULTS AND ANALYSIS

<table>
<thead>
<tr>
<th>Cases</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>slope</th>
<th>intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.8434 (+init= 0.8)</td>
<td>-0.8788 (+init=-0.3)</td>
<td>0.4027 (+init= 0.3)</td>
<td>1.6940 (+init= 0.91)</td>
<td>-0.1268 (+init= 0.05)</td>
<td>0.7851 (+init= 0)</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.47912 (+init= 0.6)</td>
<td>-0.8541 (+init=-0.3)</td>
<td>0.3850 (+init= 0.3)</td>
<td>1.3375 (+init= 0.91)</td>
<td>-0.0503 (+init= 0.05)</td>
<td>0.4300 (+init= 0)</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.3667 (+init= 0.6)</td>
<td>-1.1168 (+init=-0.3)</td>
<td>0.5605 (+init= 0.3)</td>
<td>1.4438 (+init= 0.91)</td>
<td>-0.0529 (+init= 0.05)</td>
<td>0.4652 (+init= 0)</td>
</tr>
</tbody>
</table>

**Table 7:** Table of coefficients for average stalling duration

Initial conditions, fit statistics and variables and their correlation for individual cases for average stalling duration can be given below:

<table>
<thead>
<tr>
<th>Case: 1</th>
</tr>
</thead>
</table>
| Fit Statistics :
| # function evals    | = 118 |
| # data points       | = 52  |
| # variables         | = 6   |
| chi−square          | = 136.867 |
| reduced chi−square  | = 2.975 |
| Akaike info crit    | = 62.324 |
| Bayesian info crit  | = 74.031 |

| Variables: |
| a: 0.84344791 +/- 7.624733 (904.00%) (init= 0.8) |
| b: -0.87884558 +/- 3.330198 (378.93%) (init=-0.3) |
| c: 0.40279534 +/- 1.222911 (303.61%) (init= 0.3) |
| d: 1.69408807 +/- 6.47e+06 (381979424.80%) (init= 0.91) |

| slope: -0.12681961 +/- 0.032640 (25.74%) (init= 0.05) |
| intercept: 0.78512766 +/- 6.47e+06 (824204411.87%) (init= 0) |

| Correlations: (unreported correlations are < 0.100) |
| C(d, intercept) = -1.000 |
| C(a, c) = -1.000 |
| C(a, b) = -0.999 |
| C(b, c) = 0.998 |
| C(b, slope) = 0.656 |
| C(a, slope) = -0.642 |
| C(c, slope) = 0.634 |
| C(c, d) = 0.561 |
| C(c, intercept) = -0.561 |
| C(a, d) = -0.545 |
| C(a, intercept) = 0.545 |
| C(b, d) = 0.527 |
| C(b, intercept) = -0.527 |
| C(d, slope) = 0.261 |
| C(slope, intercept) = -0.261 |
Case: 2
Fit Statistics:

- # function evals = 149
- # data points = 52
- # variables = 6
- chi-square = 18.357
- reduced chi-square = 0.399
- Akaike info crit = -42.145
- Bayesian info crit = -30.437

Variables:
- a: 0.47912898 +/- 2.470385 (515.60%) (init= 0.6)
- b: -0.85414014 +/- 1.854570 (217.13%) (init= -0.3)
- c: 0.38507121 +/- 0.728104 (189.08%) (init= 0.3)
- d: 1.33756716 +/- 7.49e+06 (560283703.22%) (init= 0.91)
- slope: -0.05034308 +/- 0.013341 (26.50%) (init= 0.05)
- intercept: 0.43001671 +/- 7.49e+06 (1742763062.89%) (init= 0)

Correlations: (unreported correlations are < 0.100)
- C(d, intercept) = -1.000
- C(a, c) = -0.999
- C(a, b) = -0.997
- C(b, c) = 0.994
- C(b, slope) = 0.510
- C(a, slope) = -0.462
- C(d, slope) = 0.447
- C(slope, intercept) = -0.447
- C(c, slope) = 0.442
- C(c, intercept) = 0.245
- C(c, d) = -0.245
- C(a, intercept) = -0.220
- C(a, d) = 0.220
- C(b, intercept) = 0.156
- C(b, d) = -0.156

Case: 3
Fit Statistics:

- # function evals = 145
- # data points = 52
- # variables = 6
- chi-square = 13.722
- reduced chi-square = 0.298
- Akaike info crit = -57.278
- Bayesian info crit = -45.570

Variables:
- a: 0.36677816 +/- 1.610197 (439.01%) (init= 0.6)
- b: -1.11683655 +/- 2.018753 (180.76%) (init= -0.3)
- c: 0.56053177 +/- 0.319216 (56.95%) (init= 0.3)
- d: 1.44381171 +/- 2.05e+06 (141912063.66%) (init= 0.91)
- slope: -0.05296105 +/- 0.008566 (16.18%) (init= 0.05)
- intercept: 0.46528049 +/- 2.05e+06 (440366719.95%) (init= 0)
Correlations: (unreported correlations are < 0.100)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C(d, intercept)</td>
<td>-1.000</td>
</tr>
<tr>
<td>C(a, b)</td>
<td>-0.999</td>
</tr>
<tr>
<td>C(a, c)</td>
<td>-0.998</td>
</tr>
<tr>
<td>C(b, c)</td>
<td>0.995</td>
</tr>
<tr>
<td>C(b, d)</td>
<td>0.609</td>
</tr>
<tr>
<td>C(b, intercept)</td>
<td>-0.609</td>
</tr>
<tr>
<td>C(a, d)</td>
<td>-0.593</td>
</tr>
<tr>
<td>C(a, intercept)</td>
<td>0.593</td>
</tr>
<tr>
<td>C(c, d)</td>
<td>0.555</td>
</tr>
<tr>
<td>C(c, intercept)</td>
<td>-0.555</td>
</tr>
<tr>
<td>C(b, slope)</td>
<td>0.465</td>
</tr>
<tr>
<td>C(a, slope)</td>
<td>-0.450</td>
</tr>
<tr>
<td>C(d, slope)</td>
<td>0.447</td>
</tr>
<tr>
<td>C(slope, intercept)</td>
<td>-0.447</td>
</tr>
<tr>
<td>C(c, slope)</td>
<td>0.424</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cases</th>
<th>PCC</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.9750</td>
<td>1.6223</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.9876</td>
<td>0.5941</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.9934</td>
<td>0.5136</td>
</tr>
</tbody>
</table>

Table 8: Results for average stalling duration

5.4.3 Total Stalling Duration

In Table 9 we see the list of coefficients and initial conditions for the Equation 3. Table 10 will give the correlation values obtained for the data. In Figure 5.31 we can see the curve fitting graph for the given equation for total stalling duration.

<table>
<thead>
<tr>
<th>Cases</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>slope</th>
<th>intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>1.1965</td>
<td>-0.8420</td>
<td>0.3896</td>
<td>1.1754</td>
<td>-0.1238</td>
<td>0.2624</td>
</tr>
<tr>
<td></td>
<td>(init= 1.2)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.6309</td>
<td>-1.3187</td>
<td>0.55611</td>
<td>1.4321</td>
<td>-0.0581</td>
<td>0.4934</td>
</tr>
<tr>
<td></td>
<td>(init= 0.8)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.7154</td>
<td>-1.5976</td>
<td>0.6491</td>
<td>1.0660</td>
<td>-0.0616</td>
<td>0.1585</td>
</tr>
<tr>
<td></td>
<td>(init= 0.8)</td>
<td>(init=-0.3)</td>
<td>(init= 0.3)</td>
<td>(init= 0.91)</td>
<td>(init= 0.05)</td>
<td>(init= 0)</td>
</tr>
</tbody>
</table>

Table 9: Table of coefficients for total stalling duration

Initial conditions, fit statistics and variables and their correlation for individual cases for total stalling duration can be given below:

Case: 1
5 RESULTS AND ANALYSIS

Figure 5.30: Curve fitting for average stalling duration
5 RESULTS AND ANALYSIS

Fit Statistics:
# function evals = 161
# data points = 52
# variables = 6
chi−square = 124.626
reduced chi−square = 2.709
Akaike info crit = 57.452
Bayesian info crit = 69.159

Variables:
a: 1.19656780 +/- 6.987841 (583.99%) (init= 1.2)
b: −0.84204705 +/- 2.070157 (245.85%) (init=−0.3)
c: 0.38969784 +/- 0.817364 (209.74%) (init= 0.3)
d: 1.17548671 +/- 3.30e+06 (280786755.33%) (init= 0.91)
slope: −0.12384986 +/- 0.033568 (27.10%) (init= 0.05)
intercept: 0.26241560 +/- 3.30e+06 (1257783314.18%) (init= 0)

Correlations: (unreported correlations are < 0.100)
C(d, intercept) = −1.000
C(a, c) = −1.000
C(a, b) = −0.999
C(b, c) = 0.998
C(b, slope) = 0.708
C(a, slope) = −0.698
C(c, slope) = 0.694
C(c, intercept) = 0.488
C(c, d) = −0.488
C(a, intercept) = −0.476
C(a, d) = 0.476
C(b, intercept) = 0.457
C(b, d) = −0.457
C(slope, intercept) = 0.414
C(d, slope) = −0.414

Case: 2

Fit Statistics:
# function evals = 138
# data points = 52
# variables = 6
chi−square = 38.366
reduced chi−square = 0.834
Akaike info crit = −3.812
Bayesian info crit = 7.895

Variables:
a: 0.63097789 +/- 0 (0.00%) (init= 0.8)
b: −1.31871846 +/- 0 (0.00%) (init=−0.3)
c: 0.55611693 +/- 0 (0.00%) (init= 0.3)
d: 1.43219619 +/- 0 (0.00%) (init= 0.91)
slope: −0.05813723 +/- 0 (0.00%) (init= 0.05)
intercept: 0.49343071 +/- 0 (0.00%) (init= 0)
5 RESULTS AND ANALYSIS

Case: 3

Fit Statistics:
- # function evals = 212
- # data points = 52
- # variables = 6
- chi-square = 17.128
- reduced chi-square = 0.372
- Akaike info crit = −45.747
- Bayesian info crit = −34.039

Variables:
- a: 0.71540998 +/− 2.464200 (344.45%) (init= 0.8)
- b: −1.59763798 +/− 2.202115 (137.84%) (init=−0.3)
- c: 0.64910146 +/− 0.147443 (22.72%) (init= 0.3)
- d: 1.06600603 +/− 1.14e+06 (107372338.17%) (init= 0.91)
- slope: −0.06165662 +/− 0.009572 (15.53%) (init= 0.05)
- intercept: 0.15852926 +/− 1.14e+06 (722011296.82%) (init= 0)

Correlations: (unreported correlations are < 0.100)
- C(d, intercept) = −1.000
- C(a, b) = −0.998
- C(a, c) = −0.967
- C(b, c) = 0.948
- C(c, intercept) = 0.712
- C(c, d) = −0.712
- C(a, intercept) = −0.531
- C(a, d) = 0.531
- C(b, intercept) = 0.482
- C(b, d) = −0.482
- C(d, slope) = 0.294
- C(slope, intercept) = −0.294
- C(b, slope) = 0.264
- C(a, slope) = −0.227

<table>
<thead>
<tr>
<th>Cases</th>
<th>PCC</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.9873</td>
<td>1.5481</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.9890</td>
<td>0.8589</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.9967</td>
<td>0.5739</td>
</tr>
</tbody>
</table>

Table 10: Results for total stalling duration
Figure 5.31: Curve fitting for total stalling duration
6 CONCLUSION AND OUTLOOK

6 Conclusion and Outlook

6.1 Recommendations

6.1.1 Service Providers

As we have seen in Section 5.1 upload speeds have little impact on initial loading delay, average stalling duration, and total stalling duration. Upload speed affects significantly only during the high download speeds. In mid-range download and upload speed, we can see more number of quality switches. To achieve better quality for given range of network parameter YouTube player uses the DASH technology and tends to switch between different qualities resulting more number of stalling. As discussed in [5], if a user faces 1% stalling ratio compared to total stalling duration is 5% likely to play the video again compared with the user who never faced stalling. Stalling has more impact on the user-behavior compared to the quality of the video is getting played. While trying to get optimized video quality users should not introduce more stalling (buffering) very often in the video. Creating many small stalling (buffering) in the video impacts the user opinion about the provided service largely. As discussed in [23], ISPs are not alone responsible for the bad video quality, sometimes the player algorithms also impact the quality of experience. ISPs are don’t get information about the player and CDN architecture of the OTTs and in many cases, OTTs also won’t be having any ideas about ISPs core network and routing algorithms. For this reason, we can not blame one player alone.

One of the recommendations to internet service providers through this project is that while providing high download links, it is also equally important to provide the high/mid-range upload links. Because low upload speeds have a higher impact on all video KQIs even in high download speeds. One of the recommendations for OTT providers is that in mid-range of download and upload speeds, it is also important to consider stabilization of video quality. For a given network condition it is better to achieve lesser stable video quality compared to racing around to many different qualities. While providing very high download speeds, it is crucial to consider upload speed also for a given network combination. It becomes significantly notable and affects user abandonment rate (simulated) by 15% even in 36 Mbps if we have a upload speed if 256 kbps.

6.1.2 QoE Measurements

It is most common practice to measure and benchmark video QoE by measuring a single video and its parameters. In Section 5.2 we have noticed that KQI ranges for measuring a single video are significantly different from measuring multiple videos in a single session. From [5] and [54] we have noted that the average video session
lasts form 24 to 45 minutes and measure only one video at a session may not be
giving the right impression of the quality experienced by the users.
This may be very significant in judging the video QoE values for OTT services
and implementation of quality models also should consider these factors which vary
for single video experience or multiple video experience. Measurement methods also
can be different for different network speeds. As we seen in Section 5.2 in lower
download speeds we can see that there is a 35% less initial loading delay and there
is a 10% reduction in the user abandonment rate if we measure multiple video session
compared to a single video session.

6.1.3 User Behaviour
As seen in Section 5.3 reloading of player helps in case of simulated users who are
actually using mid-range to high download speeds. Reloading makes it worse for
lower download and upload speeds. Also, reloading helps in reducing the stalling
ratio and user abandonment rate.

6.2 Discussion
In the project, we have shown that large-scale user-behavioral studies can also be
automated for different measurement conditions. One thing to consider in this
studies is they highly depend on the original models on which they have developed.
There comes a discussion point where we can debate on how reliable these large-scale
studies in first place. How many different parameters affect different KQIs. How
one can fully automate and create a mathematical model for these kinds of studies.
Models derived in this thesis can be used to apply in determining the different
expected video KQIs for various upload and download speeds. This thesis can be
used as a basis for measuring the user behavior of the users for different upload and
download speeds without actually measuring with the actual users.
For ISPs, one can argue that these kinds of studies can help in better utilizing
the resources without doing many economical intensive studies. They also provide
an insight on what parameter affect their users and their opinions. On the other
hand, one can also argue that to what extent these kinds of studies can actually
predict the actual user behaviors.

6.3 Summary
We tried to make a formal relation between user-behavior and different network con-
ditions. Conducting experiments to measure the different user-behavior by varying
different network conditions also became very simple and cost-effective through this
framework. Overall we can summarize this project with below points:
• Lower upload speeds during the high download speeds have high impact on all KQIs according to the graphs obtained in the results.

• In download speeds from 0.3 to 2 Mbps, there is a high variation in the number of stalling and quality count. This may be due to the player adoption strategy.

• Measuring a multiple video session compared to single video session has an impact on KQIs and during the multiple video session we get better KQI values compare to single video session.

• Reloading only reduces the startup delay in mid-range upload and download speeds. Reloading in lower bandwidth increase the initial loading delay.

• Reloading player helps for mid and high range network bandwidths.

• Quality switching happens more often in mid-range download and upload speed due to network adaptation strategy of YouTube Player.

• In case of low bandwidths, it took more time for the video to load and once it started it plays without stalling.

• Only network conditions do not directly affect the user-behavior as some of the KQIs have bad values for mid-range download bandwidth compared to lower bandwidths.

### 6.4 Future Work

This thesis serves as a basis for the network-based measurements of user-behavior. The large video logs we have collected can be later used to analyze different QoE models. This database can be used in the calculation of the MOS values for the QoE model in ITU P.1203 [17].

• 15000 video logs can be served as the starting point for network-based measurements of user-behavior. As a next step, we can start looking into the bit-stream information for each request and analysis them with respect to the video timing to understand the YouTube DASH player behavior.

• The Chrome extension can also use to measure the different content genres, length, and live videos.

• There is a need for critical analysis between Case 1 and Case 2, especially in understanding the behavior in higher bandwidths.

• This work can also be extended to study the effect of different network delays, packet loss and other network anomalies on user-behavior.
6.5 Lessons Learned

• Form this thesis one can learn different aspects of video QoE measurements and different KQIs.

• It helps in understanding the Chrome extension design and implementation.

• It helps in understanding the micro-service software architecture.

• It helps in learning python plotting libraries for data analysis.

• It helps in applying the different user-behavioral models in simulation of the different user behaviors.
References


REFERENCES


REFERENCES


REFERENCES


A.1 Box Plots for Startup Delay

Impact of Upload Speed on startup delay for varying download speeds for Case 1 can be seen here.
Figure A.1: Box plots showing the impact of upload speed on initial loading delay
A.2 Box Plots for Average Stalling Duration

Impact of Download Speed on average stalling duration for varying upload speeds for Case 1 can be seen here.
(a) For fixed upload speed of 0.256 Mbps

(b) For fixed upload speed of 0.384 Mbps

(c) For fixed upload speed of 0.5 Mbps

(d) For fixed upload speed of 0.768 Mbps

(e) For fixed upload speed of 1 Mbps

(f) For fixed upload speed of 1.5 Mbps

(g) For fixed upload speed of 2 Mbps

(h) For fixed upload speed of 3 Mbps
Figure A.2: Box plots showing the impact of download speed on average stalling duration
A.3 Plots for Showing Duration of Quality Levels

A.3.1 Different Upload Speeds

In figure A.3 we can see the bar plots showing the total time at each quality levels that are played for different upload speeds in the experiment for Case 1.
Figure A.3: Bar plots showing the total time at each quality levels for different upload speeds
A.3.2 Different Download Speeds

In figure A.4 we can see the bar plots showing the total time at each quality levels that are played for different download speeds in the experiment for Case 1.
(a) For fixed upload speed of 0.256 Mbps
(b) For fixed upload speed of 0.384 Mbps
(c) For fixed upload speed of 0.5 Mbps
(d) For fixed upload speed of 0.7 Mbps
(e) For fixed upload speed of 1 Mbps
(f) For fixed upload speed of 1.5 Mbps
(g) For fixed upload speed of 2 Mbps
(h) For fixed upload speed of 3 Mbps
Figure A.4: Bar plots showing the total time at each quality levels each download speeds
A APPENDIX

A.4 Plots for Showing Comparison of Duration of Quality Levels Between Case 1 and Case 2

A.4.1 Upload Speed of 0.256 and 3.3 Mbps

(a) For fixed upload speed of 0.256 Mbps

(b) For fixed upload speed of 3.3 Mbps

Figure A.5: Comparison of case 1 and 2 showing the total time at each quality levels

A.4.2 Download Speed of 0.256 Mbps

(a) For fixed download speed of 0.256 Mbps

Figure A.6: Comparison of case 1 and 2 showing the total time at each quality levels
A.5 Plots for Showing Comparison of Duration of Quality Levels Between Case 2 and Case 3

A.5.1 Upload Speed of 3.3 Mbps

Figure A.7: Comparison of case 2 and 3 showing the total time at each quality levels

A.5.2 Download Speed of 1.5 Mbps and 2 Mbps

Figure A.8: Comparison of case 2 and 3 showing the total time at each quality levels