UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Gestion de contenus dans les réseaux par la qualité d'expérience (QoE) : caching et transport

Othmane Belmoukadam

Université Côte d'Azur, Inria

Présentée en vue de l'obtention du grade de docteur en INFORMATIQUE
d'Université Côte d'Azur
Dirigée par : Chadi BARAKAT
Soutenue le : 27 Septembre 2021

Devant le jury, composé de :
Jérôme LACAN, Professeur, Institut Supérieur de l'aéronautique et de l'Espace
Yassine HADJAJ-AOUL, Maître de Conférences (HDR), Université de Rennes
Remi BADONNEL, Maître de Conférences, Telecom Nancy
Pedro CASAS, Senior Scientist, Institut Autrichien de Technologie
Guillaume URVOY-KELLER, Professeur, UCA
Chadi BARAKAT, Directeur de Recherche, Inria

Inria

Gestion de contenus dans les réseaux par la qualité d'expérience (QoE): caching et transport

QoE-aware content management in the Internet: caching and transport

Jury:

Rapporteurs

Jérôme Lacan, Professeur, Institut Supérieur de l'Aéronautique et de l'Espace
Yassine Hadjajd-Aoul, Maître de Conférences (HDR), Université de Rennes

Examinateurs

Remi Badonnel, Maître de Conférences, Telecom Nancy
Pedro Casas, Senior Scientist, Institut Autrichien de Technologie
Guillaume Urvoy-Keller, Professeur, Université Côte d'Azur

Directeur de thèse

Chadi Barakat, Directeur de Recherche, Inria

Abstract

The Internet has changed drastically in recent years; multiple novel applications and services have emerged, all about consuming digital content. In parallel, users are no longer satisfied by the Internet's best effort service; instead, they expect a seamless service of high quality from the side of the network. This has increased the pressure on Internet Service Providers (ISP) to efficiently engineer their traffic and improve their end-users Quality of Experience (QoE) rather than just monitoring the physical properties of their networks. Furthermore, content providers from their side, and to protect the content of their customers, have shifted towards end-to-end encryption (e.g., TLS/SSL), which has complicated even further the task of ISPs in handling the traffic in their networks. Today, the challenge is notable, especially for video streaming since it is the most dominant service and the primary source of pressure on the Internet infrastructure, imposing tight constraints on the quality of service (QoS) provided by the network. Video streaming relies on the dynamic adaptive streaming over HTTP (DASH) protocol which takes into consideration the underlying network conditions (e.g., delay, loss rate, and throughput) and the viewport capacity (e.g., viewport resolution) to improve the experience of the end-user in the limit of the available network resources [1, 2]. Nevertheless, knowing encrypted video traffic is of great help to ISPs as it allows taking appropriate network management actions. Therefore, this thesis focuses on video streaming services and video QoE to properly manage the enormous and diverse video content available on the Internet. To that aim, one needs to understand the transmission process of dynamic adaptive video streaming over HTTP (DASH) protocol, identify new metrics correlated to video QoE, and propose solutions to infer and leverage such metrics for optimal network resources management while maximizing the end-user QoE.

In the beginning, we question the efficiency of the DASH transmission in taking into account the terminal characteristics, in particular the viewport, knowing that requesting a resolution exceeding the viewport does not help much in increasing the video QoE [3]. The latter might result in a bandwidth waste, and it can either save money when the user is on a pay-as-you-go data plane or steal bandwidth from other users who need it to improve their Quality of Experience (QoE). To narrow the stats, we present a controlled experimental framework that leverages the YouTube and Dailymotion video players and the Chrome web request API to assess the impact of browser viewport on the observed video resolution pattern [4–6]. In the first attempt of this kind, we use the observed patterns to quantify the wasted bandwidth. Then, based on the viewport importance and knowing that the Internet traffic is getting encrypted, we propose a methodology based on controlled experimentation able to infer fine-grained video flow information such as chunk sizes and viewport resolution from encrypted YouTube video traces. We

leverage our dataset and supervised machine learning (ML) algorithms to train different ML models to predict viewport classes with different granularity. Later, we formulate a QoE-driven resource allocation problem to pinpoint the optimal bandwidth allocation that maximizes the QoE (Quality of Experience) for users of a network service provider located behind the same bottleneck link while accounting for the characteristics of the screens they use for video playout (viewport). Moreover, for content providers operating at the network edge, we study a viewport aware caching optimization problem for dynamic adaptive video streaming that appropriately considers the client viewport resolution and access speed, the join time, and the characteristics of videos. We propose a video content placement heuristic that balances minimal join time and maximal visual experience, subject to the cache storage capacity.

Keywords: Quality of Experience, Quality of Service, video streaming, viewport resolution, bandwidth waste, bandwidth allocation, caching, controlled experimentation, network resource management, Machine Learning, Integer Linear Programming, Non-Linear Programming

Résumé

L'Internet a radicalement changé ces dernières années, de multiples applications et services novateurs ont vu le jour, tous au sujet de la consommation de contenu numérique. En parallèle, les utilisateurs ne sont plus satisfaits du service "meilleur effort" d'Internet, mais s'attendent plutôt à un service transparent de haute qualité du côté du réseau. Cela a accru la pression sur les fournisseurs d'accès Internet (FAI) dans leurs efforts pour gérer d'une manière efficace leur trafic et améliorer la qualité d'expérience (QoE) de leurs utilisateurs finaux plutôt que de simplement surveiller les propriétés physiques de leur réseaux. Les fournisseurs de contenu de leur côté, et pour mieux protéger le contenu de leurs clients, se sont tournés vers le chiffrement de bout en bout (par exemple, TLS/SSL), ce qui a encore compliqué la tâche des ISP's vis à vis la gestion du trafic dans leurs réseaux.

Aujourd'hui, l'enjeu est notable, notamment pour le streaming vidéo qui est la catégorie d'applications la plus dominante et la principale source de pression sur l'infrastructure Internet, imposant ainsi de fortes contraintes sur la qualité de service (QoS) fournie par le réseau. En gros, le streaming vidéo repose sur le protocole de diffusion en flux adaptatif dynamique sur HTTP (DASH) qui prend en compte les conditions sous-jacentes du réseau (par exemple, le retard, le taux de perte et le débit) et la capacité de la fenêtre sur laquelle la vidéo est visualisée (par exemple, la résolution de la fenêtre d'affichage) pour améliorer l'expérience des utilisateurs dans la limite des ressources réseaux disponibles. Dans ce sens, la réalité du trafic vidéo chiffré est très importante pour les ISP's et les fournisseurs de contenu car cela leur permet d'être proactifs quand il s'agit de la gestion des flux à travers leurs réseaux (par exemple, allocation des ressources et placement de contenu). Dans cette thèse, nous nous concentrons sur les services de streaming vidéo et sur la vidéo QoE afin d'assurer une gestion efficace de l'énorme et diverse contenu vidéo disponible sur Internet. Par conséquent, il est nécessaire de comprendre le processus de transmission vidéo, d'identifier des métriques clés corrélées à la QoE vidéo et de proposer des solutions permettant de déduire ces métriques et de les exploiter afin de gérer d'une manière optimale les ressources réseaux et de maximiser la QoE des utilisateurs.

Dans un premier temps, nous allons étudier l'efficacité de la transmission DASH en tenant compte des caractéristiques du terminal, en particulier la résolution de la fenêtre d'affichage, sachant que demander une résolution dépassant celle-ci entraîne un gaspillage de bande passante. Un tel gaspillage de bande passante, si prouvé et bien maitrisé, peut économiser de l'argent lorsque l'utilisateur est sur un plan de données pay-as-you-go, et être investi auprès d'autres utilisateurs qui en ont besoin pour améliorer leur qualité d'expérience. Pour affiner les statistiques, nous présentons un cadre expérimental contrôlé qui exploite les lecteurs vidéos YouTube et Dailymotion et l'API de requête

Web du navigateur Chrome pour évaluer l'impact de la taille de la fenêtre d'affichage (en pixels, soit taille d'écran en mode plein écran) du navigateur sur la succession de résolutions vidéos observées. Dans une première tentative du genre, nous utilisons les modèles observés pour quantifier la quantité de bande passante gaspillée. Ensuite, motivé par l'impact des caractéristiques de l'écran (résolution) et la réalité du trafic Internet chiffré, nous proposons une méthodologie basée sur des expérimentations contrôlées capable de déduire des informations relatives au flux vidéo à granularité fine telles que la taille des segments à partir de traces vidéos YouTube chiffrées. Ensuite, nous exploitons l'ensemble de données collectées et des algorithmes d'apprentissage automatique supervisé pour mettre en place différents modèles capables de prédire la classe de la fenêtre d'affichage à partir des traces vidéos chiffrées. Plus tard, nous formulons un problème d'allocation de ressources piloté par la QoE pour identifier l'allocation optimale de bande passante qui maximise la QoE sur l'ensemble des utilisateurs d'un fournisseur de service réseau situés derrière le même goulot d'étranglement, tout en tenant compte des caractéristiques des fenêtres d'affichage utilisées. De plus, pour les fournisseurs de service opérant à la périphérie du réseau, nous proposons un problème d'optimisation pour la mise en cache des vidéos visualisées qui est sensible à la fenêtre d'affichage, ce dernier prend en compte la résolution de la fenêtre d'affichage des clients et la vitesse d'accès et les caractéristiques des vidéos. Par ailleurs, nous proposons une solution identifiant l'ensemble des vidéos et représentations à cacher qui assure un temps de démarrage du streaming minimal et une expérience visuelle maximale, sous réserve de la capacité de stockage du cache.

Mots-clés: Qualité de Service, Qualité d'Expérience, streaming vidéo, résolution de la fenêtre d'affichage, gaspillage bande passante, allocation bande passante, caching, expérimentation contrôlée, gestion des ressources réseaux, Apprentissage Machine, Programmation entier linéaire, Programmation non-linéaire

Acknowledgements

This is the end; after a thrilling three years, I cannot even find words to express my feelings and gratitude toward all those people who supported me during my Ph.D. First of all, I thank God, he has graced my life with opportunities that I know are not of my hand or any human hand. I particularly thank my supervisor, Professor Chadi Barakat, for his unlimited help, guidance, and support on so many levels, without which I wouldn't have started a Ph.D. and accomplished it. Chadi is a great person, both personally and professionally, he was always present to support me in my periods of stress. He helped me work on topics that I like and pushed me to explore new ideas and tackle challenging problems. Having the opportunity to work with him both in my Master's degree and my Ph.D. is a privilege that I will always be proud of. I thank the reviewers for dedicating time to review my thesis and thank the examiners for accepting to be part of my defense jury. A warm thanks to all the members of the DIANA team, for the unforgettable moments at Inria Sophia Antipolis. In particular, Naoufel, Abdelhakim, Mohamed, Jawad, Houssam, Anouar, Yanis and all the other fellow Ph.D. students I spent most of the time drinking coffee discussing relevant and irrelevant topics; I will never forget those moments.

I would also like to thank my family for the support they gave me throughout my Ph.D. Without a doubt, whatever I am today is because of my beloved parents. They sacrificed and endured difficult moments to provide everything so that I can pursue my dreams. Their prayers, moral and material sacrifices remain the source of my success. To my brothers, thank you for encouraging me to always go forward. To my uncles and close friends, thank you for believing in me and providing guidance when needed. Moreover, I am incredibly grateful to a particular person in my life, and I would like to thank her for showing me the true meaning of care, support, and sacrifice despite the circumstances.

Finally, this Ph.D. was an unforgettable journey full of sweat, stress, laughter and tears. On many occasions, I looked myself in the mirror and thought of quieting; luckily, for all the above reasons, I made it through and learned that limits like fear are often just an illusion.

Contents

List of Figures

List of Tables

Abbreviations

QoS	**Q**uality **o**f **S**ervice
QoE	**Q**uality **o**f **E**xperience
ML	**M**achine **L**earning
MOS	**M**ean **O**pinion **S**core
RMSE	**R**oot **M**ean **S**quared **E**rror
HAS	**H**TTP **A**daptive **S**treaming
DASH	**D**ynamic **A**daptive **S**treaming over **H**TTP
ABR	**A**daptation **B**it**R**ate
ILP	**I**nteger **L**inear **P**roblem
NLP	**N**on **L**inear **P**roblem
DPI	**D**eep **P**acket **I**nspection
HD	**H**igh **D**efinition
SD	**S**tandard **D**efinition

Dedicated to my family and beloved ones

Chapter 1

Introduction

The Internet has seen tremendous structural changes throughout the years (Figure 1.1). It started at the very beginning with the need to create a vast computer network. Afterward, the (TCP/IP) technology was proposed, a resilient and robust protocol stack able to link several networks together such that, if one network is down, the others do not collapse [11]. Later, the Internet started providing essential services such as connecting multiple hosts, sending emails, and sharing files. In the 90s, the Hypertext Markup Language (HTML) was invented by the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG), giving birth to the first incarnation of the World Wide Web [12]. After that, several companies started infiltrating the market, with Microsoft launching Windows 95, Amazon, Yahoo, and eBay launched, and Java created, thus allowing for animation on websites and opening the door for new internet activity. Today, the Internet is revolutionizing people's life, from shopping to ordering food, sharing moments with family and friends, ending with instant messaging. Everything is one click away from anybody anywhere on the globe.

Today, video traffic is unarguably the major contributor to global Internet traffic and the main source of pressure on the Internet infrastructure. By 2023, video traffic is expected to account for 73% of the global mobile data traffic, compared with only 56% in 2015 (Figure 1.2) [7]. Moreover, due to the COVID-19 pandemic, sanitary confinement forced people around the globe to restrict their mobility and increase their video traffic for remote working, entertainment, and education. Recently, researchers have analyzed data from ISPs, IXPs, and educational networks to show that video traffic has increased by 15-20% almost within a week [13]. A statement by the European Union raised concerns about the coronavirus lockdown putting strain on broadband delivery systems. As a result, mainstream content video providers, such as Netflix, reduced their video resolution to the standard definition during the pandemic [14]. Afterward, providers

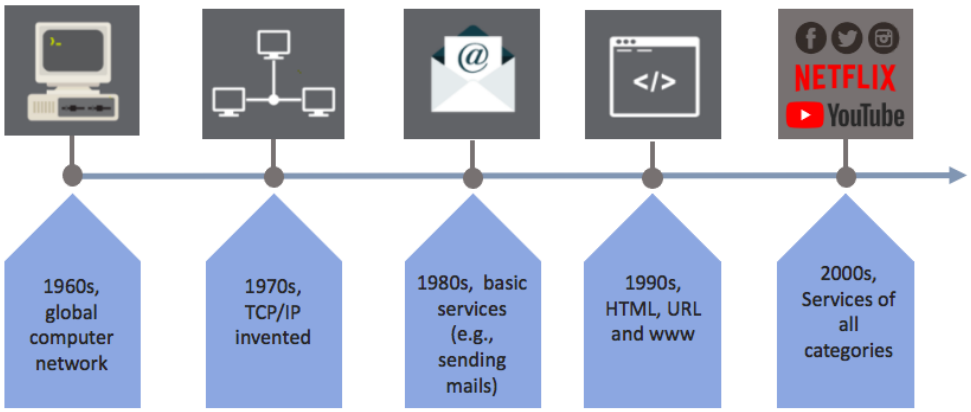


FIGURE 1.1: The main Internet milestones

started again to upgrade their services back to high definition [15]. The end-users from their side expect the best quality and can be frustrated by any service interruption. Statistics show that up to 90% of end-users abandon their operator after experiencing network quality degradation without giving any feedback, resulting in substantial economic losses. As a result, both operators and service providers feel more pressure to be proactive and enhance their services as much as possible by assessing the end-user Quality of Experience (QoE).

Quality of Experience (QoE) can be defined as the level of satisfaction of a customer's experiences with a service. It can be characterized by a set of factors, some are subjective while others remain objective such as the network conditions and the terminal display capacity [16]. By definition, the QoE is a subjective metric; however, it can be evaluated by subjective, objective and hybrid methods. The subjective QoE evaluation requires a panel of humans that will grade their experience after watching a set of videos. The ITU-R BT.500-1 [17] is a standard method for conducting subjective video quality evaluations. On the other hand, the objective QoE evaluation methods leverage raw metrics that can be computed with technical parameters collected from the network. We mention Peak-Signal-to-Noise Ratio (PSNR), an objective QoE evaluation metric which is defined as the Mean Squared Error (MSE) between an original frame and the distorted frame [17]. The PSNR can be computed only when the image is downloaded by the end-user, thus, limiting its usage in real-time scenarios. Meanwhile, Pseudo-Subjective Quality Assessment (PSQA) is a hybrid QoE assessment approach, combining advantages of both the objective and subjective schemes [18]. They call it hybrid because it incorporates a subjective evaluation in its methodology performed only once to calibrate a model that can be used many times as necessary to input the objective parameters. In the context of video streaming over wireless networks, authors in [19], demonstrate

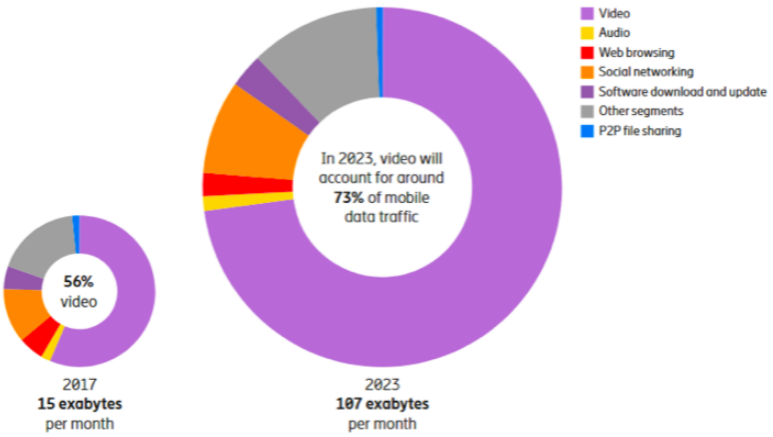


FIGURE 1.2: Mobile video traffic statistics, figure from [7]

that the PSQA approach out-perform the QoE estimation by the objective Peak-Signal-to-Noise Ratio (PSNR) metric and gives a similar result comparing to subjective QoE values given by real-users.

In this era where end devices such as mobile phones, tablets, and monitors, are dotted with more advanced features (e.g., the viewport resolution), the device-related factors (e.g., screen resolution) play a significant role in defining the visual experience. In plain, the perceptual visual experience is highly affected by the screen display characteristics (size and resolution) and human influence factors such as the viewer distance from the screen and its visual acuity. The visual acuity measures the sharpness of vision so that people with 20/20 vision can see clearly at 20 feet. On the other hand, people with 20/200 vision must be at 20 feet to see what a person with normal vision can see at 200 feet. Moreover, it has been proved that if the pixel size is inferior to the smallest visual angle at which two separate objects can be discriminated (minimum separable angle), the structure of the pixel is invisible and does not negatively affect image quality [20]. In this context, and considering viewers of normal visual acuity (20/20), sitting at a standard distance from the screen (not too close as the field of view gets wide, and not too far as the field of view becomes fogy), the screen characteristics mainly its resolution plays a significant role in defining the perceptual visual experience. With that being said, we still need to understand the impact of the screen characteristics (e.g., resolution) on adaptive video streaming. In particular, to what extent modern video transmission techniques account for the screen resolution before deciding which video resolution to request from the server. Such a study can also shed light on any form of bandwidth waste related to downloading resolutions exceeding the screen resolution (i.e., a viewport in full-screen mode). This waste can be of particular concern to end-users paying their

subscription at the byte level and to operators who can invest it on other flows in need of it.

Motivated by the increasing demand for video streaming services, video content diversity, and the different display technologies in the market, we tackle three main challenges in this thesis that can be summarized as follow:

1. The interaction between the video delivery and the end-user viewport resolution in adaptive video streaming
2. The key features of the encrypted video traces that can be used to infer information such as the viewport resolution
3. The management of video content by leveraging QoE models accounting for the viewport resolution, both in terms of resource allocation (e.g., bandwidth sharing) and caching (e.g., content placement)

In the next section, we discuss the challenges and briefly present the solutions proposed in the thesis.

1.1 Challenges AND Motivation

1.1.1 Video transmission techniques

Today, and almost on every platform, video streaming is governed by the Dynamic Adaptive Streaming over HTTP (DASH) protocol. For DASH, the client player automatically switches between video resolutions according to underlying network performance [1, 2]. The video chunk resolutions requested from the server are determined by the network conditions estimated by the DASH client. They usually reflect the viewport resolution, which is defined as the number of pixels, vertically and horizontally, on which the video is displayed. In scenarios where bandwidth is scarce, either because of congestion or limited bandwidth at the access, the impact of the screen is expected to be negligible. However, when the network is in good conditions, the user's fair share of bandwidth might exceeds the viewport requirements, which raises the question of whether the players stick to the viewport requirements or exceed it to download more than their needs. Downloading any resolution exceeding the viewport will be automatically downsized, hence resulting in a waste of bandwidth. This waste can go unnoticed in cases of abundant bandwidth, however in cases when bandwidth is billed at the byte level (the pay-as-you-go data planes) or network is saturated, understanding and controlling such waste can lead to a

less economical loss for the end-user and improved Quality of Experience (QoE) for the other users who are in need for it (e.g., users with large viewports).

The challenge is thus notable when studying the video transmission process from a device display perspective. First of all, we need to consider the diversity of video content (slow motion, high motion, and static) and stream videos from different types and categories. Moreover, we need to account for different screen resolutions as the market proposes plenty of technologies. To answer this challenge, we propose an experimental framework to stream thousands of videos automatically from different types and categories on different viewport resolutions. Our framework leverages the YouTube and Dailymotion media players and the Chrome web request API to assess the influence of browser viewport on the observed video resolution pattern [4–6]. To the best of our knowledge, this is the first experimental study highlighting how far the browsers today consider the viewport resolution. Furthermore, we use this study to approximate the bandwidth waste in video streaming applications, resulting from downloading video resolutions exceeding the end-user display capacity.

1.1.2 Video QoE modeling

When addressing Quality of Experience in general, modern standards (e.g., ITU) focus on protocols or general classes of applications and not on specific Internet services. Among these, most target application-level metrics, such as the number of interruptions or the number of bitrate switches during a video playback [21, 22] or the signal degradation within VoIP services [23]. For what concerns the network aspects and how they affect the user Quality of Experience (QoE), standards give general recommendations (e.g., consider only one parameter or define minimum thresholds to have service). For specific Internet services (e.g., YouTube), researchers either focus on the application or the network-level QoS metrics. For instance, Hoßfeld et al. study how the Youtube QoE is affected by application-level QoS measurements [24]. YoMoApp [25], a video streaming crowdsourcing application, targets the collection of Youtube's QoS metrics and users' feedback. Authors in [26] perform QoE forecasting using machine learning models taking as input network-level QoS metrics (e.g., throughput and delay) collected by end-user devices with the help of active measurements. Following the Pseudo-Subjective Quality Assessment (PSQA) approach, Kandaraj et al. propose a QoE monitoring module which uses Random Neural Networks (RNN) and estimates the QoE using statistics related to video playout interruptions (e.g., number, average delay and maximal delay) along with a quantization parameter that controls the amount of video compression [27].

In a nutshell, researchers generally focus on the application-level QoS (e.g., stalls and resolution switches) or the network-level QoS metrics (e.g., throughput, loss rate, and delay) when addressing video QoE. However, the visual aspect of video QoE is also vital to consider as it is mainly related to the device and human-related factors. From our side, we enlighten the relationship between video QoE and the viewport resolution. We use controlled experiments to study the impact of the viewport resolution on the video resolution patterns. We also push the video QoE modeling of state of the art by considering QoE models that reflect the trade-off between the screen resolution, the video bitrate, and the video QoE. Our QoE models fit an exponential function that maps QoS and QoE parameters. We follow the IQX hypothesis, an exponential interdependency between QoE and QoS metrics, it takes as input network QoS parameters to determine the QoE, and we extend it to different viewport resolutions corresponding to watching modes supported by video providers [28].

1.1.3 Video traffic encryption

Nowadays, customers and even official governments raised many concerns about data privacy. Following public demand, and to avoid sanctions, content providers have shifted toward end-to-end encryption (e.g., TLS/SSL) with more than 50% of the Internet traffic getting encrypted [29]. With little visibility on the Internet traffic, particularly the video traffic traversing their network, Internet service providers cannot do much in terms of QoE-aware resource management. Given both the encryption of video traffic and the different metrics impacting video QoE, capturing the latter faithfully can be very challenging. In this context, we decide to study the encrypted video traces, and engineer features that hint inferring the end-user viewport resolution.

For the sake of clarity, and regardless of the encryption, we highlight different types of QoS measurements related to video traffic:

- **In-band features**: this set includes features related to the video application adaptation to the network conditions. For instance, the chunk sizes download and its variation, which reflects the throughput or buffer size estimation by the DASH client (depending on the rate adaption algorithm [30, 31]).
- **Out-of-band features**: this set includes features such as bandwidth, jitter, loss rate, and delay. In a controlled environment, these measurements can be orchestrated and reproduced based on previous traces. They can be emulated in the lab by network emulators such as Linux traffic control (*tc*).

In this thesis, we study encrypted video traces and mainly focus on the **In-band** network features such as the chunk size, the throughput, and the packets' inter-arrival time. We are the first ones to propose a data-driven methodology based on controlled experimentation and machine learning (ML) to produce ML models able to predict the viewport resolution from the YouTube encrypted traces.

1.1.4 QoE-aware resource management

As mentioned earlier, mobile operators and network providers prioritize QoE as a metric to assess users' satisfaction and avoid any economic loss. For the same goal, 5G networks promise high connectivity and huge transmission capacity aiming to take the internet services and the corresponding user experience to the next level [32, 33]. Moreover, network slicing allows the Internet to host separately different services (e.g., video streaming, smart city, and autonomous driving), each having its resource requirements that need to be provisioned. As a result, the management of Internet resources is a very interesting and challenging task to solve.

After studying the importance of the device's display capacity regarding the transmitted video chunks, consumed data, and video QoE, we propose a solution to infer the viewport resolution from the video encrypted traces. Then, we work on QoE models that consider both device-related and media or network-related factors. To complete the puzzle, we leverage our QoE models to solve two main problems related to resource management on the Internet:

1. **Bandwidth sharing**: given the problem of resource allocation for several video flows sharing the same bottleneck, we develop an optimization framework to maximize the overall QoE by optimally sharing the available bandwidth while taking into consideration terminal display capabilities (viewport resolution).

2. **Caching**: given the problem of cache placement in adaptive video streaming, we develop an optimization framework that jointly accounts for the end-user display capacity and video characteristics (e.g., encoding bitrate and popularity) in addition to the internet access speed to decide on the video content to cache at the edge of the network. Our solution provides the set of video representations with the optimal QoE gain and optimal policy for adding video representations if storage is available.

1.2 Thesis Roadmap

Here, we summarize the outline of this thesis:

1. In Chapter 2, we overview the literature on video streaming in terms of protocols and transmission techniques. We revisit existing video QoE models and explain the QoS to QoE modeling methodology. Then, we highlight the main contributions related to video QoE inference from encrypted traffic. Finally, we discuss how researchers have accounted for QoE models when performing resource-sharing optimization.

2. In Chapter 3, we start by highlighting the diversity of video content in the Internet, and we study the popularity and bitrate encoding of different video categories for two video providers, YouTube and Dailymotion. Considering this video diversity, we study video content delivery with the modern DASH protocol while focusing on the impact of the end-user display capacity (viewport resolution). Our experimental framework leverages the YouTube and Dailymotion video players along with chrome browser API and a large set of videos from different types and categories [5]. We analyse the impact of the viewport resolution on the observed video resolutions and use it to approximate the bandwidth waste related to downloading resolutions exceeding the display capacity of the end-user device.

3. In Chapter 4, and given the importance of the viewport resolution, we propose an experimental methodology based on controlled experiments and machine learning to infer the viewport resolution from the YouTube encrypted video traces. Our study highlights a strong correlation between the viewport resolution, the chunk sizes, and the throughput. In plain, our approach starts with streaming thousands of YouTube videos extracting features that correlate to the viewport resolution (e.g., chunk size) from the passive video traces. It ends up with training/testing machine learning models that can predict the viewport resolution with different granularity using input statistics calculated over our set of **In-band** features.

4. In Chapter 5, we explain our methodology for building QoE models that account for device, media and network-related factors. Our QoE models use open-source datasets to fit an exponential QoE function and map either throughput or video bitrate to a QoE level [9, 10] for a set of different viewport resolutions. Then, we use these QoE models to solve a bandwidth sharing problem for a set of users streaming videos over the same bottleneck link. Further, we develop a simple and greedy heuristic based on Lagrangian multipliers and use ns-3 to implement it and compare it to legacy solutions. The network simulations show that our solution

results in better overall QoE and enhances application-level QoS metrics such as the average number of stalls and the average resolution switches.

5. In Chapter 6, we shift our interest to caching and content placement. We formulate the optimal cache placement problem for adaptive video streaming in a way to allow caching multiple representations of the same video. The proposed cache placement algorithm leverages the users' viewport resolution and decides on the videos and the representations per video to cache based on an objective function reflecting the QoE relation to the video content (bitrate), the application-level QoS (join time), the viewport resolution, and the access speed distribution. Overall, we develop a heuristic called *QoEscoreMax* that ranks video representations based on expected QoE values and decides which one to cache. Further, our heuristic can help content providers derive the optimal policy when caching multiple representations of the same video.

6. In the last chapter, we conclude our work and present aspects that can be further improved in future work.

Chapter 2

State of the Art

In this chapter, we discuss different aspects related to video streaming and video Quality of Experience (QoE); starting with *(i)* the video transmission process in adaptive video streaming, followed by *(ii)* the approach used to measure and model video QoE, up to *(iii)* the inference of video QoE indicators from encrypted traffic, and ending with *(iv)* the QoE-aware network resource management.

2.1 Video streaming background

As of 2020, the worldwide video streaming market size was estimated at 50 billion (USD). The latter is projected to have a compound annual growth rate (CAGR) of 21% from 2021 to 2028 [34]. For this critical Internet service, the ecosystem is simple; the end-user puts the URL of a streaming platform and click on the desired video to watch. Behind the screen, content providers take charge of the delivery and storing of video content. After clicking, the video starts downloading, and once a buffer threshold is reached, the video starts playing. The end-user keeps downloading video parts while the previous ones play out.

In the early 90s, Windows and Apple launched their Media player and Quick, respectively, with reading and playing audio and video files. Later, companies like Netflix started investing in video streaming through their DVD rental websites. At that time,

people were still relying on physical video rental stores like Blockbuster. In the early 2000s, the YouTube video streaming platform started, and Amazon initiated what is now known as Amazon Prime. Later, with Google acquiring YouTube, Netflix Watch Now, and Hulu debut, video content is one click away from anybody anywhere worldwide. Recently, with the pandemic and mobility restrictions, the users' demand for video streaming services is as high as ever, and it exceeded all expectations. To that aim, all stakeholders of the video entertainment industry and even service providers are urging to create their streaming platforms starting with HBO streaming service owned by AT&T and WarnerMedia, up to Disney+ the streaming platform hosting all Disney content. For latter video streaming services, called **Video on Demand (VoD)**, the videos will be stored and organized by a media distribution system that allows users to watch without any fixed broadcast schedule. Today, pre-payed VoD services such as Netflix and Amazon prime are top trending services on the Internet. The latter allows the purchase and even rental of video content. On the other hand, **Live Streaming**, is when the video is streamed in real-time. Most social networking applications enable live streaming to stream and interact with others; in this category, one can also find video calls.

Most of today's content providers support both VoD and Live streaming. In this work, we consider the case of VoD through some of the leading video content providers, name YouTube and Dailymotion.

2.1.1 Video transmission techniques

Nowadays, both content providers and network operators give high importance to video streaming services. The Internet best effort, designed with non-real-time data transmission considerations, raises several challenges regarding the delivery of video content while preserving the best Quality of Experience (QoE). To overcome the high demand, several approaches have been proposed through the years. The latter can be clustered into two subsets, as one leverages the HTTP protocol and the other does not. We refer to protocols and delivery paradigms that do not incorporate the features of the HTTP

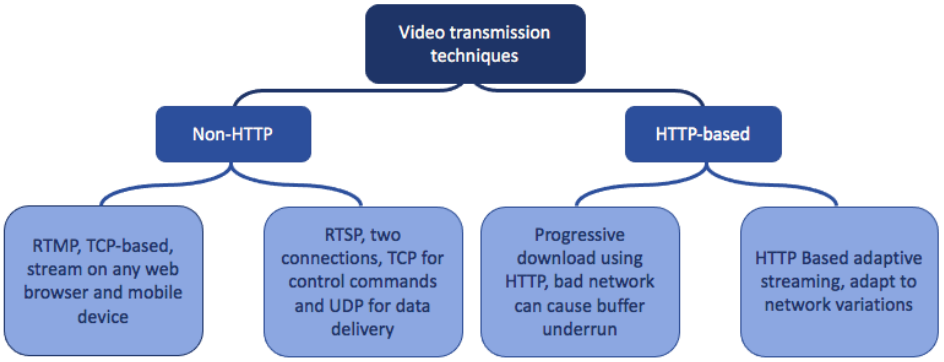


FIGURE 2.1: HTTP-based and non-HTTP video delivery techniques

protocol as non-HTTP. It follows that the other media delivery schemes will be named HTTP-based[1].

2.1.1.1 Non-HTTP based video delivery

For non-HTTP media delivery systems, the end-users receive audio or video content from a media server using protocols such as the Real-Time Messaging Protocol (RTMP). Historically, RTMP is the reason why live streaming is possible. It started as a tool to transmit content between a video player and a server, referred to previously as RTMP delivery. Lately, with HTTP Live Streaming (HLS) taken its place, RTMP became responsible for transmitting video files from an encoder to a video hosting platform. RTMP is connection-oriented as it relies on the Transmission Control Protocol (TCP) (Figure 2.1). By default, it is secure and able to transmit multimedia on any mobile device and web browser. Meanwhile, Real-Time Streaming Protocol (RTSP) surges as an application-level protocol managing different data types delivery with real-time properties. RTSP enables on-demand delivery of real-time data such as audio and video. The sources of data can include both live data feeds and stored clips. This protocol is originally maid for multiple data delivery sessions, and it can be supported by both transport-level protocols TCP and UDP. It can also provide means for choosing delivery mechanisms based upon RTP [36]. RTSP uses two connections, the TCP connection handles the control messages, including the stream's commands, while UDP handles the

[1]Figure 2.1, classification of some video delivery protocols and their main features [35].

data delivery (e.g., video and audio) (Figure 2.1). Despite the benefit of minimal delays and fast transmission, using UDP might result in video frames distortion or dropping due to packet losses.

2.1.1.2 HTTP-based video delivery

Meanwhile, almost every modern streaming system is using the HTTP protocol for delivering video content. The HTTP protocol is easy to deploy and secure, with the HTTPS version ensuring data encryption when exchanging content between a web browser and a website [37]. In terms of transport protocols, HTTP supports TCP as well as QUIC, which means that video content over HTTP is reliably transferred to the client, and there can be no frame dropping, hence, minimizing video distortion [37, 38]. This leads to the discussion of video transmission over HTTP and its development through the years; it started with videos downloaded completely by the clients before being played out before streaming platforms started allowing progressive video delivery to the clients at a fixed resolution. Later, **progressive streaming over HTTP** took place. In this context, the video is divided into small pieces called segments or chunks. The end-user clicks on the desired video to watch and automatically starts filling the buffer. This latter has two thresholds: the maximum one being the occupancy limit and the minimum one being the threshold above which video starts playing out. If the clients experience network condition degradation, eventually, the buffer will dry out, and interruption thus occurs. We are technically below the minimum buffer threshold. The playback resumes as soon as new video segments are downloaded.

Recently, **HTTP adaptive streaming (HAS)** has been widely adopted to automatically tune the streamed video resolution as a function of the available network resources. For instance, Dynamic Adaptive Streaming over HTTP (DASH) protocol [1, 2] allows adapting the video quality to the available bandwidth and the client terminal characteristics (e.g., viewport resolution). In plain, DASH divides the video into segments (e.g., 2 - 10 seconds), with each segment available in different quality versions. Details on the video availability at the server are stored in the Media Presentation Description (MPD) template, which describes the video segments in terms of coding standard and bitrate

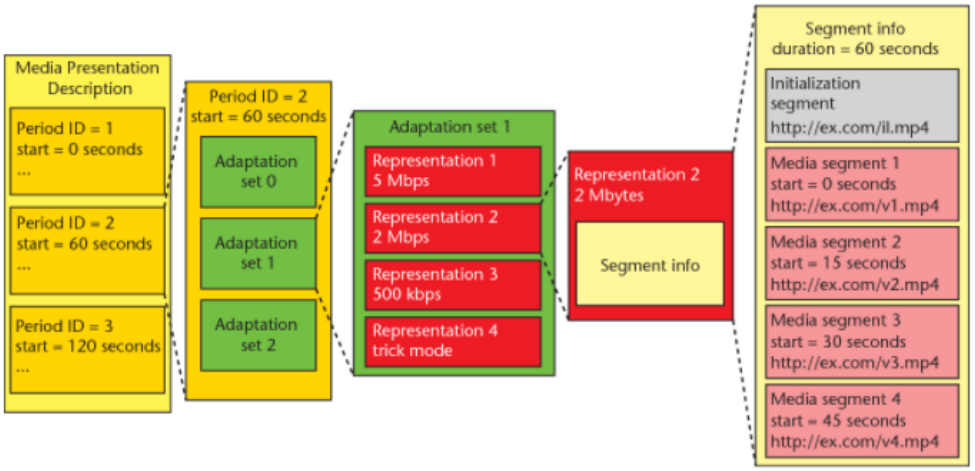


FIGURE 2.2: Media Presentation Description sample, figure from [8]

and is shared with the client at the beginning of every video session. In Figure 2.2, we illustrate in detail the hierarchy of the MPD file. At the top of the hierarchy, we find the different periods with their corresponding identifiers and timestamps. For each period, a set of adaptations is defined. These adaptations describe the different representations of the media components with their bitrate encoding. They can correspond to video media components or audio media components. For the same representation, the MPD file lists the metadata of each of its segments, and such metadata contains the ID, the starting time, and the location on a remote server [8]. The different segments can be downloaded using the HTTP GET message with byte ranges (as highlighted in the HTTP clear text messages in Chapter 4).

The choice between the different video representations is made by the DASH client, with as objective the smoothest possible playout without excess bandwidth usage [35]. In Figure 2.3, we highlight this process briefly, with the DASH server hosting the different representations of each video. Each video has its specific MPD file. The DASH client starts by downloading the MPD file and controls the choice of the next chunk to download based on the client network conditions (i.e., bandwidth, delay, and loss) and usually has to consider the characteristics of the displaying screen [8].

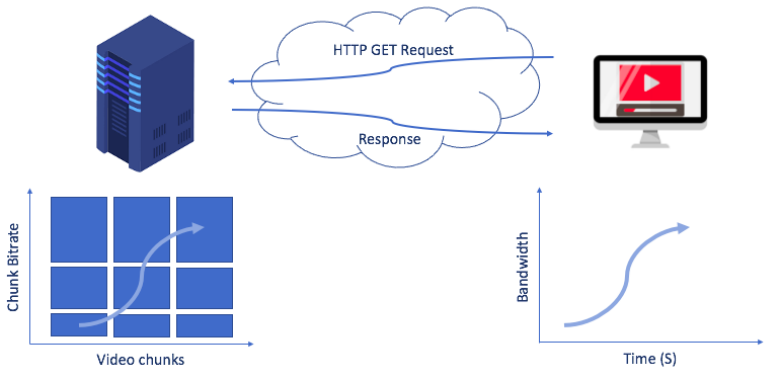


FIGURE 2.3: HTTP adaptive streaming in a nutshell

2.1.1.3 Adaptation BitRate (ABR) algorithms

As mentioned earlier, DASH divides the video into small pieces of equal duration, with each piece available in different bitrates. The mechanism to select the following resolution to download is an essential tool to optimize the end-user QoE. The technique to choose the appropriate chunk resolution that maximizes the video QoE in light of the undergoing network conditions is called the ABR algorithm. Inventing algorithms for optimal ABR is an active area of research, with multiple papers proposing different approaches [38]. The rate adaptation algorithms remain the most common approach for video QoE optimization, especially at the application-level. Other solutions operating at the network-level or even at the edge of the network will be discussed in more detail in subsection 2.4.

Next, we classify the ABR algorithms and discuss their main fundamentals:

1. **Throughput-based ABR**: the throughput-based ABR algorithm continuously estimates the end-user throughput, from which the majority of variants derive fixed rules regarding the bitrate of the next video chunk to download from the server [39, 40].

2. **Buffer-based ABR**: due to throughput fluctuations over time, throughput-based algorithms are more likely to cause video resolution switches. To overcome this problem, researchers suggest using only the buffer size. In this context, the video rate is based on the current buffer occupancy [41, 42].

3. **Hybrid ABR**: the combination of both buffer and throughput feedback is an interesting solution. The two signals complement each other as using simple capacity estimation (based on immediate past throughput) can help enormously during the startup phase, when the buffer itself is growing up from empty [43].

4. **Learning based ABR**: the main limitations of state-of-the-art ABR algorithms can be summarized in adapting fixed control rules based on simplified models. As a result, the previous approaches fail to achieve optimal performance across a broad set of network conditions and QoE objectives. Learning-based solutions leverage deep learning techniques to generate ABR algorithms. They learn to make bitrate adaptation decisions through observations of the resulting performance of past decisions [44].

Up to this point, we revisited the background on video streaming. We highlighted the mechanisms of adaptive video streaming, mainly the different components of its ecosystem and how they interact. To that aim, the user perception of this service needs to be continuously monitored through the user perception of this service needs to be continuously monitored through the Quality of Experience (QoE) of end-users. In the next section, we walk through the different approaches addressing video QoE.

2.2 Video QoE modeling

As highlighted in the Introduction, the QoE is a subjective metric assessing the level of users' satisfaction regarding an application or a service. In particular, the video QoE represents the viewer's perception of a video streaming service. The **subjective video QoE evaluation** is based on panels of actual users by explicitly rating their perceived QoE either to classes with good/bad labels or via a rating on a continuous scale. In the latter case, the score usually ranges from 1 to 5, where 1 corresponds to the lowest perceived QoE and 5 to the highest perceived one. The average of all the collected ratings is what we call the Mean Opinion Score (MOS), a QoE metric standardized by the ITU-T P.910 recommendation [45].

The subjective evaluation methods have directly correlated the application-level QoS metrics and the Mean Opinion Score (MOS). Several researchers managed to develop MOS prediction models that map the QoS measurements to a MOS value based on their findings. For instance, the IQX hypothesis highlights an exponential correlation between Quality of Service (QoS) metrics and QoE [28]. Further, Hossfeld et al. use the subjective approach and real-users to highlight a logarithmic impact of the join time on the perceived QoE [46]. Authors in [24] highlight the same exponential relationship this time for between the number and duration of video stalls (i.e., interruptions) and the MOS is exponential. These models can be considered pseudo-subjective in the sense that there is the human factor at a certain point, the subjective aspect can be either at the modeling level through previous studies that have shown a specific pattern when correlating QoS metrics to the MOS (i.e., exponential or logarithmic) or explicitly by using real-user to calibrate the model once and then using it as many as necessary with as input the objective metrics (i.e., the ITU P.1203 and ACQUA [22, 26]).

On the other hand, the **objective video QoE evaluation** does not incorporate the human aspect, and it uses automated methods and technical indicators to evaluate the video quality and emulate the human visual system (HVS). The Peak-signal-to-Noise Ratio (PSNR) is the most common objective metric to evaluate the spatial quality of videos, it consists of computing the level of distortion between the original, and the received image [47]. The PSNR compares the maximum possible signal energy to the noise energy image by image. Authors in [47], define a heuristic mappings the PSNR to MOS value, as a PSNR of more than 37 (dB) and less than 20 (dB) result in A MOS of 5 and 1, respectively. Another objective video quality assessment metric is the Structural SIMilarity (SSIM) index; the SSIM uses the sliding window approach to calculate the structural distortion of an image instead of error-sensitivity-based metrics (i.e., PSNR). The SSIM index does not require spatial or temporal filtering nor linear transformations, it can be implemented in real-time schemes, and the image sampling rate can be tuned to increase its speed [48]. The no-reference objective metrics analyze the test video and do not need any information about the origin video, thus, more suitable for real-time video quality assessment. Among the no-reference objective video metrics, we find the blockiness, which evaluates the artifacts of compression methods such as MPEG [49].

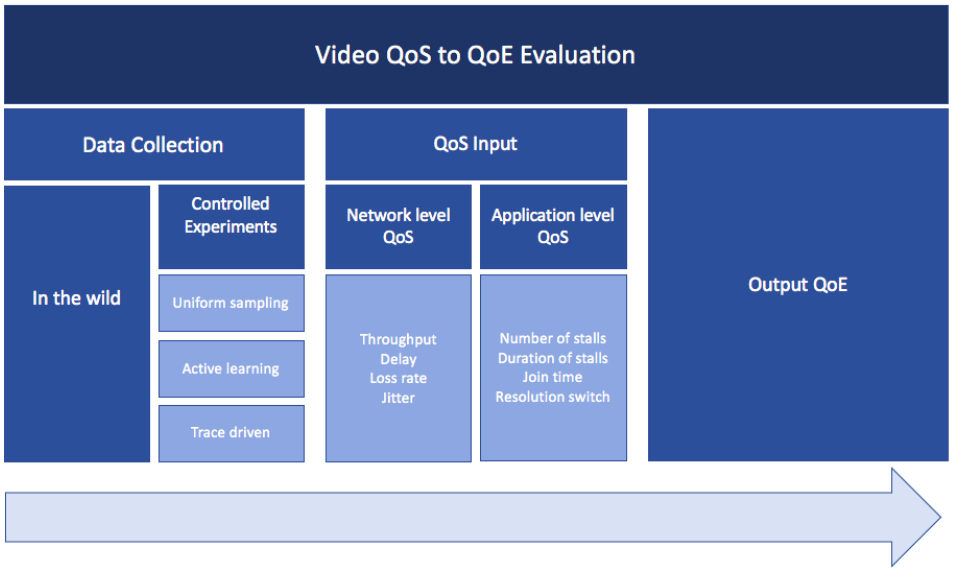


FIGURE 2.4: The video QoS-to-QoE modeling process

In general, the video QoS-to-QoE modeling consists of building the mapping between the application-level or network-level QoS metrics and the video QoE. From our perspective, this modeling process can be divided into three main steps; (i) the data collection as most of the works related are data-driven, (ii) the QoS input type as the QoE of video streaming is dependent on the application QoS metrics, which in turn are dependent on the network QoS metrics, (iii) the evaluation method. In Figure 2.4, we summarize the typical steps for video QoS-to-QoE evaluation. Each of the following subsections revisits in detail the different steps and their corresponding sub-approaches.

2.2.1 Data Collection

As in the case of other data-driven approaches, robustness and preciseness require a large pool of samples. The latter data can be collected either;

From the **wild**, crowdsourcing or panels, real-users give explicit satisfaction values directly or via their end-devices or by using the measurements collected by service providers. Recent mobile applications provide an easy and cost-efficient crowdsourcing solution as they can reach a large group of people and collect much more data in

short time periods [25, 26, 50]. For instance, YoMoApp, an android application monitoring passively the application-level video playout metrics (i.e., stalls and resolution changes) for YouTube QoE from end-user smartphones [25].

From **controlled experiments**, the video is played out in a controlled environment where the network conditions can be emulated. The controlled experiments enable video streaming at a large scale in an automated manner that does not require the interference of real-users. Authors in [51], use controlled experiments and play several videos in different network conditions. They use this data to train Machine Learning (ML) models able to predict the QoE of a video stream using as input the video characteristics (i.e., video bitrate) and the network-level QoS measurements. Meanwhile, different sampling methodologies exist to tune the network parameters and explore the ample space of network conditions. Among these sampling approaches, we mention the default random and uniform, which can be time-consuming, especially if working with different network metrics (e.g., bandwidth, delay, and loss rate) [52]. Another standard sampling methodology consists of reusing traces or what is called trace-driven sampling. Yitong et al., illustrate a subjective video quality assessment with real-users under trace-driven network emulations for adaptive video streaming sessions [53]. Recently, active learning is a new methodology applied to QoS to QoE modeling, which samples the experimental space intelligently to reduce the training cost [54].

In this thesis, we use controlled experiments to stream videos at a large scale while considering the diversity of video content, different viewport resolutions, and multiple network settings. From our side, in Chapter 4, we use random sampling along with video bitrate traces to sample the bandwidth and enforce it using the Linux traffic controller (*tc*) [55].

2.2.2 QoS input type

The QoS-to-QoE video models are calibrated to take the QoS input and turned it into a QoE indicator describing the client's level of satisfaction. The QoS input type can be different from one model to another as the literature focus on two subsets:

2.2.2.1 Network-level QoS

In this category, we find the network-level QoS metrics related to the **in-band** QoS measurements that can be collected from the video traffic. The network-level QoS metrics such as the throughput are indeed responsible for the quality of the video chunks requested (see throughput based ABR 2.1.1.3) and therefore impact the video resolution quality. The packet loss, another metric impacting the playout metrics as higher packet losses result in video distortions and therefore impacting the perceived user experience heavily [26, 56, 57]. Generally, the Internet Service Providers (ISP) can access and approximate these metrics and monitor the QoE of their end-users.

In the literature, several papers tweak these network parameters via controlled experimentation and study the impact of the network conditions on the perceived application-level QoS metrics. In [9], severe degradation of the throughput or loss rate can result in several video interruptions and resolution switches, impacting the perceived QoE negatively. Furthermore, authors in [58] present ML models that can infer the video playout metrics such as the number of stalls and the resolution of the played video within a window of 10 seconds using the network and the transport level features of the encrypted YouTube video traffic. The last study highlights the correlation between QoE indicators that can be easily noticed by the viewers and the undergoing network conditions.

2.2.2.2 Application-level QoS

In this category, we define mainly QoE metrics related to the video playout. These metrics manifest at the application-level, and end-users can notice them directly from the video playback, which is not the case for the previous ones.

1. The video **startup delay:** also known as the join time, it can be defined as the difference between the timestamp of the click on the video and the timestamp for the beginning of the playout. The joining time is in tight correlation to the video QoE; studies have shown its impact on user abandon rate. According to authors in [59], users start abandoning the video session after 2 seconds of join time, and 80% of them leave the session when their join time exceeds 60 seconds.

2. The video **stalls:** the stalls or video interruptions happen when the buffer dries out. In the literature, many papers discuss the impact of video interruptions in terms of number, duration, and even localization over the video session. Authors in [60] prove the link between large stalls duration and video QoE decrease. Moreover, they show that end-users prefer one long stalling event over frequent short ones. On the same topic, researchers have shown that video interruptions positioned at the beginning of the playout have a lesser impact than those that occur later [61].

3. The video **resolution switches:** the adaptive video streaming is made to adapt the video resolution to the network conditions. Meanwhile, a client can choose a fixed low resolution to avoid interruptions in case of bad network conditions. However, when relying on dynamic adaptive streaming, too many quality changes can decrease video QoE [62]. Authors in [63] have shown that end users would appreciate a video session at a low quality rather than a video session with high quality but interrupted with many resolution switches.

2.2.3 The output QoE

The third aspect of the QoS-to-QoE modeling process is related to the output QoE and the video QoE assessment methodology. After collecting the data and choosing the QoS input type, the final step consists of labeling the data or aggregating the collected metrics as an indicator that can approximate the perceived video QoE. At this stage, researchers either set up panels of real-users that will watch videos in the sampled network conditions and then grade their perceived video experience (i.e., MOS from 1 to 5) [17]. This approach is the closest to capture the human visual system (HVS) but requires real users continuously, which is time-consuming and does not scale well given the diversity of video content and the multiple combinations of network settings that we can explore.

The pseudo-subjective method consists of using functions or models fitted with subjective QoE data from real users. These models can map objective technical metrics like the network-level QoS or the application-level QoS to a pseudo-subjective MOS at

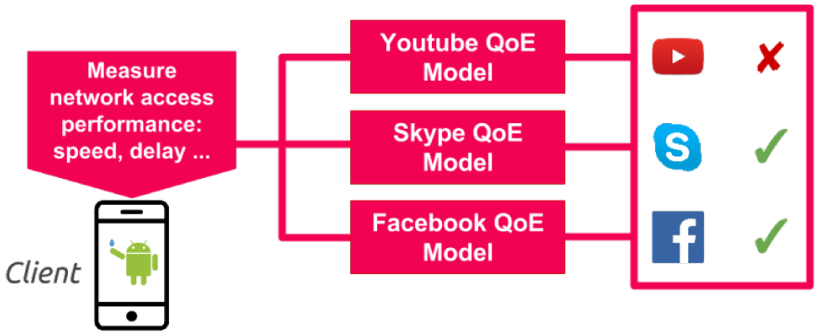


FIGURE 2.5: ACQUA objective QoE modeling

their final state. The ITU-T P.1203 is a parametric and automated model for audio-visual quality assessment of adaptive video streaming calibrated with subjective Mean Opinion Scores (MOS). Relying on MOS, the ITU-T P.1203 can be viewed as a pseudo-subjective QoE evaluation method; it incorporates different blocks from the video quality estimation module to the audio quality estimation module up to the quality integration module. This model has different operation modes depending on the input features. By default, the primary operation mode (i.e., 0) takes as input the metadata from the video segments. Other modes require additional information on the frames and coding standards [21]. The ACQUA application developed by the DIANA team at Inria Sophia Antipolis is another example of pseudo-subjective QoE modeling tools. ACQUA relies on active measurements to collect network QoS metrics from end-user devices. These metrics are used to perform controlled experimentation and stream different videos, objective QoE metrics such as stalls and join time can be calculated and aggregated to one QoE meter using the ITU-T P.1203 model. Then, ACQUA uses machine learning models to link internet access conditions to an estimated QoE [26]. In Figure 2.5 we highlight ACQUA's QoE modeling process and the principle of QoE forecast. In general, QoE forecasting consists of reusing the same network QoS input to get a QoE estimation for different applications, each of which we find the video on demand through YouTube.

For the sake of completeness, in Table 2.1 we highlight a set of applications specialized in network monitoring and QoE assessment, and we show their strengths and weaknesses. Overall, most solutions available do not generally offer QoE insights but only focus on the technical details (i.e., network QoS metrics). Moreover, popular applications in the domain of network sensing and QoE assessment, e.g., [64–68], use similar methodology

to estimate the network conditions. These techniques leverage ICMP ping for delay and loss measurements and TCP-based downloads/uploads for bandwidth measurements. While being very practical and precise, these techniques, especially the TCP-based one, consume tens of megabytes per measurement session. Thus, these applications usually leave the user manually triggering each measurement and do not propose a periodic measurement plane. Among all the approaches we found in state of the art, only Sensorly [68] seems to be fully tailored for continuous mobile network sensing by allowing users to perform network measurements in the background. While being adapted to this use case, Sensorly uses TCP-based speed tests to estimate the bandwidth and consumes a considerable amount of data when not in passive mode.

Tool	Mobile	Light	QoE	Active/passive	Output
iPerf	No	No	No	Active	Network QoS
SpeedTest	Yes	No	No	Active	Network QoS
MobiPerf	Yes	No	No	Active	Network QoS
Sensorly	Yes	No	No	Active	Network QoS
Meteor	Yes	No	Yes	Active	Multi-service QoE
RTR-NeTest	Yes	No	Yes	Active/Passive	Friendly meters
YoMoapp	Yes	Yes	Yes	Passive	YouTube QoE
QoE Docotr	Yes	Yes	Yes	Passive	user latency
ACQUA	Yes	Yes	Yes	Active/Passive	Multi-Service QoE

TABLE 2.1: QoE and network sensing tools comparison

To summarize, video QoE is an exciting and challenging topic. Many researchers have shown its dependence on application-level QoS and network-level QoS metrics. Meanwhile, this era is characterized by multiple display technologies, which give the device display capacity a more significant impact on the video QoE. The literature is still missing an extensive study of the device display influence on the perceived watching experience (through the video resolution patterns) and the data consumption. If this impact is highlighted, the video QoE modeling methodology needs to account for the device factors and the other exploited metrics. First, we propose a complete study based on real experimentation to investigate how far the browsers today take into consideration the display capacity and quantify the bandwidth waste in video streaming applications.

2.3 Video QoE and encrypted traffic

The traffic identification from encrypted traces is an active field of study. Methods based on Deep Packet Inspection (DPI) offer solutions to inspect and take actions based on the payload of the packets rather than just the packet header. Machine Learning (ML) is widely exploited in the DPI field [69–71]. ML algorithms proved their efficiency, learning from big data and statistical properties of the traffic flow. However, these algorithms pass by a heavy training phase and might struggle in processing complexity if run in real-time. Another well-known DPI technology is OpenDPI, which is freely available and includes the latest DPI technology combined with other techniques making it one of the most accurate classifiers nowadays [72]. Khalife et al. [73] attempt to reduce the OpenDPI computational overhead by examining different sampling techniques. Two sampling techniques are proposed and compared: *(i)* per-packet payload sampling, and *(ii)* per-flow packet sampling. Enhancing DPI performance is as active as inventing new DPI technologies, so several approaches have been proposed, including behavioural [74], statistical [75], port-based [76] and DFI (Deep Flow Identification) based approaches [77]. Other approaches apply software-based optimization focused on enhancing DPI algorithms, e.g., [78], while other approaches rely on hardware-based optimisation [79].

In the context of end-to-end encryption and knowing the reality behind video QoE, researchers leverage encrypted traffic by passively monitoring the network and capturing traffic statistics that are then transformed into video QoE. For instance, we find work on inferring video interruptions, video quality, and quality variations by observing network-level traffic statistics [80]. Others use a large number of video clips to identify specific Netflix videos leveraging only the information provided by the TCP/IP headers [81]. Dimopoulos et al. [80] propose to use the size of video chunks as input to machine learning. However, their method requires access to the end-user device to collect real values about these chunks instead of inferring them from the encrypted packet traces. They also provide the first heuristic to automatically extract chunk size information from encrypted traffic based on identifying long inactivity periods along with the video streaming session. Silhouette [82], a video classification method, uses Application Data

Units (ADUs) and network statistics to detect and infer properties about video flows. The method leverages downlink/uplink packet characteristics to identify chunk requests and corresponding information sent by the server (chunk size). It only incorporates static thresholds making it unable to differentiate between video and audio chunks.

While effective in terms of video traffic identification, the cited contributions focus on inferring the network or media-related video QoE indicators, yet they overlook the device-related factors that can drastically impact the perceived video QoE. Within this limitation, our experimental study of the screen resolution and its impact on the adaptive video streaming process highlights further the importance of the display capacity in terms of video resolution patterns and data consumption. To complement our work, we propose a methodology to infer the end-user viewport resolution from the encrypted traffic, which, together with the information on the video flow such as the streaming resolution and the application-level QoS, can provide the ISP a fine-grained estimation of the user QoE.

2.4 QoE-aware resource management

To manage the content available on the Internet properly and leverage the plethora of QoE models provided by the research community, several researchers propose optimization frameworks leveraging QoE models and promising the enhancement of the user experience, yet, each approach operates on a different level:

2.4.1 Network-level optimization

The topic of QoE-driven resource sharing has already been investigated on several occasions. For example, in terms of routing, neural networks have been used in wired and wireless networks to optimize QoE. Previous works try to select the best path using network-level QoS features as QoE replacement (e.g., loss rate, delay) [83, 84]. Mao et al. propose using reinforcement learning (RL) to generates ABR agents. Their solution, called *Pensieve*, leverages observations collected by client video players to train a neural network model that selects bitrates for future video chunks. The authors show

that *Pensieve* outperforms the best state-of-the-art schemes, with improvements in average QoE [44].

Quang et al. illustrate QoE-driven routing as a MILP problem by considering Pseudo-Subjective Quality Assessment (PSQA) as a QoE model and propose a heuristic solution [85]. Moreover, Calvigioni et al. consider the HTTP adaptive streaming (HAS) flow requirements and study them in conjunction with TCP. They use a linear QoE function to express a joint routing and resource allocation problem and propose a dual sub-gradient approach based on Lagrangian relaxation sub-problems to select a single best path upon each request [86]. In another work [87], the authors express a rate allocation problem to maximize a two-term power series model over three requested resolutions and under link capacity constraints. The optimal solution is implemented in switches through weighted fair queuing and by using OpenFlow. However, the utility function used depends on the characteristics of a test video that is too specific and less generic since it requires a mapping per video at each resolution. Moreover, using the Structural Similarity Index (SSIM) as a metric to assess quality requires particular state sharing with the controller, which can be tricky giving the prevalence of encryption-based delivery. In [88], the authors present another approach based on queuing, without routing or any client modification. The latter paper proposes a controller able to track the clients' buffer states and prioritize queuing for the flow in danger of interruptions. In MPEG-DASH SAND, one can find the coupling of network assistance with coordination from the client. The network has a general view and can provide bandwidth reservation at routers; clients are in charge of the final decision and can receive recommendations [89][90][91]. However, studying the resource-sharing problem from an end-device perspective and with trace-driven models for video QoE remains an open and challenging problem. Therefore, we focus on video streaming and build QoE models to capture the link between video bitrates or throughput and QoE for different viewport resolutions (screen resolution in full-screen mode). We define a resource sharing problem to maximize the sum of the non-linear QoE functions under linear constraints on the screen resolution and the bottleneck link utilization. We also validate our model and compare the different solutions with a realistic DASH implementation in ns-3.

2.4.2 Caching

The above approaches can only be adopted by network operators, who can prefer anything but changing the core network. The idea of caching video content as close as possible to the end-user has thus emerged as an attractive alternative solution for both ISP's and content providers. Mobile edge caching thus became a very active field of study. Authors in [92] discuss thoroughly the benefits and limitations of caching content at the wireless edge and the needs to be considered for designing cache placement strategies. They also introduce methods to predict the popularity distribution and user preferences. Always in the wireless context, researchers have proposed the use of small cells called "helpers" to add caching functionality at the cellular access. The *femtocache* approach, for example, incorporates a wireless-distributed caching network that assists the base station by handling requests of popular files that have been cached, thus minimizing the download delay of users [93]. The *femtocache* approach only considers the video popularity and the network conditions. The work in [94] formulates a joint routing and caching problem aiming to maximize the fraction of content requests served locally by the deployed small base stations (SBS). In this reference, the joint optimization considers the bandwidth capacity constraints of the small cell base stations. However, it overlooks the video content characteristics. Sengupta et al. propose an architecture to identify popular multimedia content by proactively pushing it close to the edge of the wireless network, thereby alleviating backhaul load [95].

At the same time, adaptive video streaming is known to increase the challenge of edge caching as each video can be available in multiple representations. In this scope, several caching schemes have been proposed to leverage caching for dynamic video streaming. Zhang et al. [96] propose a caching scheme to maximize the QoE for end-users under a limited storage budget. To this end, a logarithmic model for the QoE was used, leading to a constrained convex optimization problem. Authors in [97] propose iPac, an integrated prefetching and caching proxy that maximizes the byte hit ratio in the context of limited bandwidth between proxy and content server. The iPac proxy does not require any modification to existing content servers and video clients. Jin et al. [98] leverage the cloud and the assumption that end users can only access one edge server at

a time to examine a three-way trade-off between caching, transcoding, and bandwidth cost on each edge server in a way to reduce the total operational cost. In particular, they analytically derive the closed-form solution of the optimal transcoding configuration and caching space allocation for each edge server [98]. Gao et al. leverage users' viewing patterns and propose a cost-efficient transcoding scheme to balance transcoding and cloud storage [99]. Last but not least, in [100], authors investigate the bitrate oscillations resulting from the presence of a cache server on the path between the DASH client and server. They propose an approach to reduce these oscillations by adaptive controlling the rate at which the clients download video segments from the cache, resulting in a smoother video playout.

To summarize, the information on the QoE is essential and has mainly been used to cope with the exponential growth of Internet video traffic. However, the literature is still missing a study that accounts for the viewport resolution and its heterogeneity across the viewing users when it comes to video caching. Such heterogeneity, coupled with the heterogeneity of the network access speed, and the fact that videos are available in different bitrates, impact cache placement; choosing the representation(s) to cache is still an open and interesting problem to solve. This constitutes the main focus of Chapter 6.

2.5 Novel contributions

To the best of our knowledge, this thesis features the following novel contributions. First, we present an experimental study highlighting the interplay between the device display resolution and the video content delivery in adaptive video streaming. We show that the DASH clients might request video resolutions exceeding the one of the viewport, hence, resulting in a bandwidth waste with no significant impact on the perceived visual experience. At a macro level, there is a negative correlation between the viewport resolution and the bandwidth waste. Meanwhile, this waste might be different from a content provider to another depending on their media player implementation.

We also present a controlled experimental framework that helped us identify features incorporating the signature of the viewport resolution via only encrypted video traces. We followed a data-driven approach with machine learning to classify the viewport resolution using input statistics calculated over in-band network-level features inferred from encrypted traces.

Lastly, we study Internet resource management with the help of QoE models while accounting explicitly for the device-related factors, along with media-related (e.g., video bitrate) and network-related factors (e.g., throughput). We leverage in particular QoE models and viewport resolution to optimize the bandwidth sharing for a set of users streaming videos over the same bottleneck link and propose heuristics to select the most QoE rewarding video representations to cache in the case of adaptive streaming.

Chapter 3

On the impact of the viewport resolution in adaptive video streaming

Video streaming is, without a doubt, the most dominant application on the Internet. Each time a video streaming platform (e.g., YouTube, Dailymotion, or Netflix) is requested, the browser loads a web page, sets up the video player, retrieves and renders the requested content. The video streaming transmission is based on the dynamic adaptive streaming over HTTP (DASH), which considers the underlying network conditions (e.g., delay, loss rate, and throughput) to select the video resolution requested from the server. We question in this chapter the efficiency of this transmission in taking into account the terminal characteristics, the viewport resolution, in particular, knowing that requesting a resolution exceeding the viewport results in a waste of bandwidth. The latter bandwidth can either save money when the user is on a pay-as-you-go data plane or steal bandwidth from other users who need it further to improve their Quality of Experience (QoE). In the first attempt of this kind, we present a controlled experimental framework that leverages the YouTube and Dailymotion video players and the Chrome web request API to assess the impact of browser viewport resolution on the observed video resolution pattern [4–6]. Later, we use the observed pattern to quantify the amount of wasted bandwidth.

3.1 Introduction

Today, and almost on every platform, video streaming is governed by the DASH protocol. For DASH, the client player automatically switches between video resolutions according to underlying network performance [1, 2, 101–103]. As requested from the server, the video resolution pattern is thus determined by the network conditions captured by the DASH client. It has normally considered the viewport resolution, which is defined as the number of pixels, both vertically and horizontally, on which the video is displayed. In scenarios where bandwidth is scarce, either because of congestion or limited bandwidth at the access, the impact of the screen is expected to be negligible. However, when the network is in good conditions, the user's fair share of bandwidth exceeds the viewport requirements, which raises the question of whether the players stick to the viewport requirements or exceed it to download more than their needs. Downloading any resolution exceeding the viewport will be automatically downsized, hence resulting in a waste of bandwidth. This waste can go unnoticed in cases of abundant bandwidth, however in cases when bandwidth is billed at the byte level (the pay-as-you-go data planes) or network is saturated, understanding and controlling such waste could lead to less economic loss for the end-user and improved Quality of Experience (QoE) for the other users who are in need for it (e.g., users with large viewports).

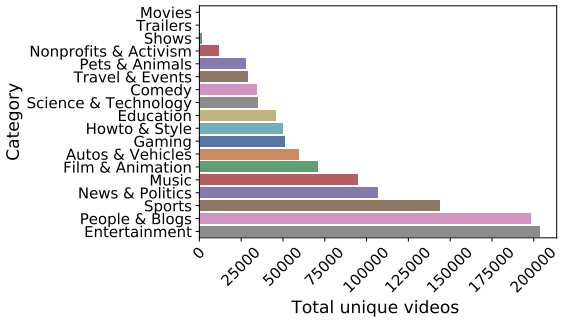
In terms of QoE, and as stated in Chapter 2, the video QoE is affected by video stalls, join time, and resolution switches which are all related to the achieved bandwidth by the video flow. Cermak et al. [10] answered the question concerning the bandwidth needs for acceptable video experience on a set of screen resolutions partially. They show that different screen resolutions have different bandwidth requirements for the same QoE level. Other researchers have worked on QoE-driven network optimization, providing solutions both at the network and the application-level [44, 85, 86]. For instance, and among many other related work, routing of video flows is optimized in [85, 86] to improve the QoE of end-users, whereas the adaptation of video streaming quality is optimized in [44] using Deep Reinforcement Learning for smoother playout and improved end-user QoE.

To the best of our knowledge, we are the first to propose a complete study based on real experimentation to investigate how far the browsers today consider the viewport resolution and quantify the bandwidth waste in video streaming applications. For this, we build web pages that embed video players for two main streaming platforms (YouTube and Dailymotion). Every time called, the video player automatically gets a random video ID and a specific viewport then starts playing the video while we collect measurements from within the browser. We leverage the Chrome web request API to read the HTTP explicit texts and record on a remote database the video chunks information [5]. We then use this information to derive models for the video resolution pattern on different viewports for both YouTube and Dailymotion and use these patterns to identify the waste of network resources and estimate the amount of this waste. Overall, the contributions of this chapter are:

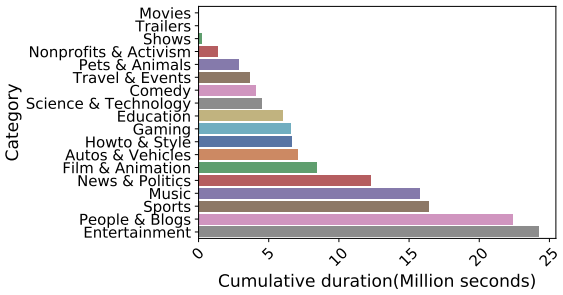
- We provide an overview of today's YouTube and Dailymotion video catalogs (e.g., popularity, video formats, and bitrate). To that aim, we rely on an open-source dataset of YouTube video metadata [104]. For Dailymotion, we build our own catalog of trending videos covering several categories, and we make it available to the large public [105].

- We present a controlled experimental methodology to identify the video resolution patterns on different viewports. Our methodology is general and can be used to extend the work to streaming platforms other than YouTube and Dailymotion as long as these platforms provide video player and data API. Moreover, our framework accounts for the high variability of video content by considering a large YouTube catalog and several Dailymotion playlists.

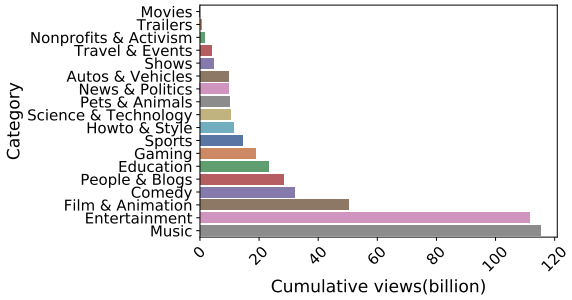
- We provide a detailed performance comparison for YouTube and Dailymotion players with a focus on the bandwidth waste on a set of viewports. For YouTube, we collect the HTTP requests to highlight the chunk resolution pattern [4, 5]. For Dailymotion, we propose a probabilistic methodology able to quantify the bandwidth waste based on the real-time periodic player updates.



(a) Cumulative number of videos per category



(b) Cumulative video duration per category



(c) Cumulative view count per category

FIGURE 3.1: YouTube catalogue overview per category

The rest of this chapter is organized as follows. In Section 3.2 we provide a descriptive analysis of both catalogs used in our experiments. In Section 3.3 we discuss the architecture of our framework for the two considered streaming platforms. Later in Section 3.4, we discuss the data-driven video resolution patterns discovered using different Chrome browser viewports for YouTube and Dailymotion. Then, in Section 3.5, we leverage the underlying patterns to approximate the resulting bandwidth waste. Last, we summarize the main contributions of this chapter.

3.2 Video content overview

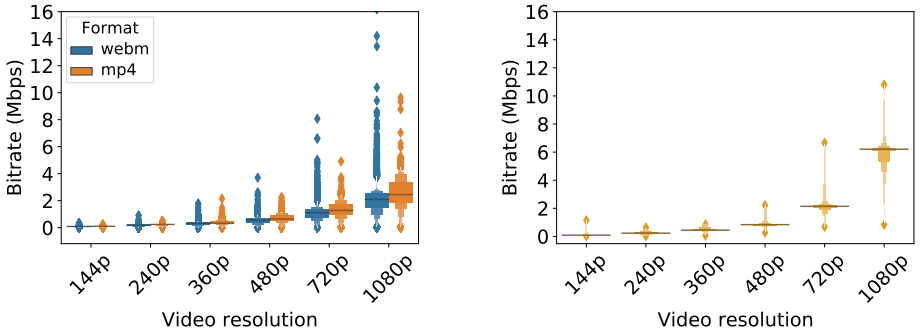
We use an open-source YouTube catalog, the catalog was built using the YouTube API, where YouTube was searched with specific keywords obtained from Google Top Trends website. The authors of [9] rely on Google's *getvideoinfo* API to return the video metadata for each video identifier. The dataset includes around 1 Million unique video identifiers. For Dailymotion, we fetch 200 trending videos from different categories. Regarding the diversity of content in the dataset, we observe that the YouTube videos belong to several categories: sports, entertainment, and gaming. In Figure 3.1, we plot statistics regarding YouTube video categories. As can be seen in Figure 3.1(a), the entertainment and people & blogs categories are the largest ones representing each more than 200K unique videos, then followed by the sports category with almost 150K unique videos. On the other hand, the least represented categories are movies and trailers with less than 10K videos. In terms of duration, the people & blogs videos are the longest with a cumulative duration of 8 Million seconds, followed by the entertainment category with a total of 6 Million seconds (Figure 3.1(b)). At the bottom of the list, we find movies with a cumulative duration of fewer than 1 Million seconds. We also evaluate the popularity of each category in terms of the number of views. For this, we aggregate the total views for videos belonging to the same category and show the results in Figure 3.1(c). As highlighted in the figure, the entertainment and music videos are the most popular with a cumulative number of views almost equal to 120 Billion each, followed by film & animation with up to 60 Billion views. At the bottom of the list, one can find movies and trailers with less than 1 Billion views each. Unfortunately, we couldn't perform the same study for Dailymotion because of the lack of the corresponding metadata and the difficulty of measuring it at a large scale. However, to give an idea of trending categories on Dailymotion, we refer to the statistics on most viewed channels available in [106].

3.2.1 From video resolution to bitrate

The adaptive video streaming requires different video resolutions, each characterized by an encoding bitrate that differs from one video to another depending on its content

(high motion, slow motion, no motion or static, music video, ...). It also differs from one resolution to another for the same video. The video bitrate is an essential feature in our study that allows estimating the bandwidth waste. In terms of video distribution per resolution, 99% and 60% of the videos featured by YouTube and Dailymotion, respectively, support video resolutions up to 1080p. For YouTube, the remaining 1% support higher resolutions (e.g., 2160p and 2880p). Due to the small proportion of the latter resolutions, we limit our study to videos available in multiple resolutions up to 1080p. Thus, we study the bitrate distribution for main video resolutions, ranging from 144p to 1080p. In the YouTube catalog, videos are available in two major video types encoded by the H.264 and Google's VP9 standards[107, 108]. The individual video type formats are "mp4" and "WebM", respectively. By analyzing the catalog, we found that 82% of the videos are available in mp4 and WebM. The remaining 18% of videos are only available in mp4. To study the difference between the two formats, we illustrate their bitrate distribution using boxen plots, enhanced versions of box plots featuring several quantiles, and offering more details while describing empirical distributions.

The overall distribution of the video bitrate w.r.t. the supported resolutions and video types is given in Figure 3.2(a). The two video formats have slightly different bitrates for the same resolution with an advantage for the WebM format, making it the preferred format by Google to handle the video content bulk. Overall, and as expected, the figures show a clear positive correlation between video resolution and the bitrate. On the other hand, for Dailymotion, we use a video downloader to obtain the actual size of all videos used in our experiments for every available resolution. We then calculate the reference bitrate per resolution as being the video size divided by the video duration. Note that our method does not distinguish between audio and video packets. They both contribute to the video size. This can still be considered a good approximation of the real video bitrate, especially given the low standard audio bitrate highlighted in the Dailymotion official documentation [109]. We highlight in Figure 3.2(b) the bitrate distribution w.r.t. video resolution for Dailymotion videos. In general, Dailymotion shows the same macro behavior as YouTube, where the encoding bitrate increases with the video resolution. These values highlighted in our study correlate with the results mentioned in the Dailymotion official documentation [109]. By comparing the two plots



(a) YouTube video bitrate (webm vs mp4) (b) Dailymotion video bitrate (mp4)

FIGURE 3.2: Video bitrate per resolution for YouTube and Dailymotion

in Figure 3.2, we can see a clear difference in bitrate values between the two platforms, especially for the 1080p resolution. Overall, YouTube features lower encoding bit rates for high resolutions hence suggesting more efficient utilization of network resources for videos of the same resolution. Such difference could be partially related to the encoding parameters, such as the frame rate. Further, YouTube video encoding shows more variability in the bitrate compared to the Dailymotion one.

For YouTube and the comparison between the WebM and mp4 video formats, our observations on the bitrate are in line with the prior study in [110] on the correlation between the two formats for encoding multimedia content and the user experience. Indeed, the authors in [110] compare the two formats from the perspective of the Mean Opinion Score (MOS), highlighting an advantage of the H.264 (mp4) when the network conditions are favorable, while the VP8 codec (WebM) behaves better in highly error-prone networks.

3.3 Experimental setup

We plan on highlighting the video resolution pattern played on a given Chrome browser viewport. Normally, this pattern is affected by the viewport, but also by the underplaying network conditions [16, 60, 111]. As we are focusing in this study on the viewport and the extent to which it is respected by the player, we exclude the network impact by only experimenting with good network conditions able to support the best resolutions

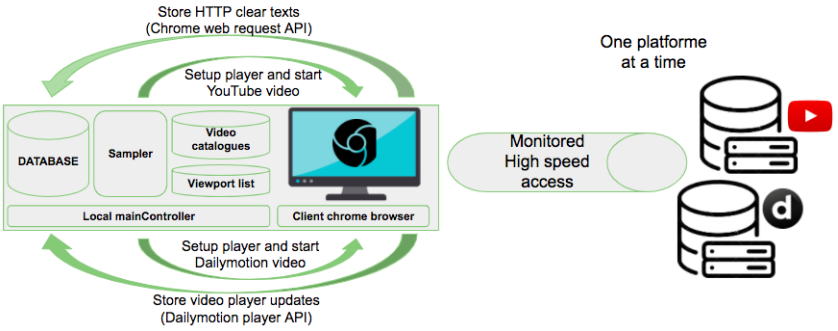


FIGURE 3.3: Experimental framework description

available for each video. Regardless of the streaming platform, good network conditions in our framework consist of a wired connection ensuring high download bandwidth monitored to never go below 10 Mbps. This choice is also motivated by the bitrate distributions for both YouTube and Dailymotion (see Fig. 3.2). Our overall experimental setup described in Figure 3.3 consists of a local *mainController* running on a MacBook Air machine of 8 GB RAM. Videos are visualized on a Dell screen 27' of 2560 x 1440 resolution. The local *mainController* stores the video catalogs and the viewport list and provides a random combination of video ID and viewport for every new experiment. We cover a large space of viewports by considering a list of default standard viewports such as the current YouTube and Dailymotion small media player mode (400x225) along with other default SD viewports (e.g., 240x144, 640x360, and 850x480). These latter viewports represent the current player dimensions adopted by streaming platforms for several watching modes. We also account for high definition viewports by considering the 1280x720 and 1920x1080. In fact, as of March 2021, stats show that up to 70% of desktop screens worldwide are of resolution less than or equal to 1920x1080 [112].

3.3.1 YouTube use case

For YouTube, and to study the chunk resolution pattern on a given viewport, we stream up to 2000 YouTube videos covering different categories (e.g., sports, entertainment, education) and a large span of video bitrates. First, we use the iframe player API to embed a YouTube video player on our web page and control the player using JavaScript functions [4]. While playing a video, the audio and video chunks are requested using separate HTTP requests, each with a specific resolution. We leverage the Chrome web

request API to extract the HTTP clear requests and get the chunk-related information (e.g., itag). To interpret the chunk itags, we use the YouTube open documentation, which is publicly available and which allows mapping itags to chunk resolution and coding standard [113, 114].

3.3.2 Dailymotion use case

Along with YouTube, we consider the Dailymotion video-sharing application. Dailymotion is available for Windows 10, Windows Phone, iOS, and Android mobile operating systems, and most recently for the PlayStation 4 and Xbox One gaming consoles. From one side, we aim at confronting YouTube to Dailymotion in terms of their video resolution pattern for different viewports and the amount of bandwidth wasted on each viewport. From the other side, we will get to understand the particular interaction between the Dailymotion video player and today's interactive web pages. While the HTTP messages can be intercepted following the previous methodology applied for YouTube, no documentation is available to shed light on the raw metadata included in the messages, making it hard to interpret in terms of resolution and coding standard. To overcome this limitation, we propose a probabilistic methodology to estimate video resolution patterns using real-time updates from the Dailymotion player accessed through its own API. In general, regardless of the streaming platform, the player API does not give access to chunk resolutions neither chunk sizes. Instead, it can be used to access player properties such as current playback time and resolution. The same API can be used to collect application-level QoS features such as stalls and join time, which is very useful for QoE monitoring.

So, we use the Dailymotion JavaScript SDK (Software Developer Kit) to embed their video player in our web page and access all its features [115]. Among the features available, we capture the real-time player updates on downloaded resolutions and export them to our database, where they are stored for later processing. The *mainController* highlighted in Figure 3.3 performs the main offline tasks of setting up the Dailymotion player within a specific viewport (same viewport list as in YouTube experiments) and launching the streaming of one of the considered videos. However, instead of intercepting

Video resolution	Viewport (pixels)
1080p	1920x1080
720p	1280x720
480p	850x480
360p	640x360
240p	426x240
144p	240x144

TABLE 3.1: Standard video resolution with matching viewport

HTTP clear requests, as in the case of YouTube, we periodically report to our database the player updates in a real-time fashion.

3.4 The impact of the browser viewport on the video resolution patterns

In this section, we leverage our controlled experimental framework (see Figure 3.3) to conduct a data-driven analysis of video resolution patterns observed on different browser viewports. We assume that each resolution results in the best visual experience when displayed on the corresponding viewport (i.e., an equal number of pixels) without any stall and with a reasonable start time [3]. To motivate this assumption further, we leverage an ITU-T Rec. P.1203 standalone implementation with an open-source dataset based on controlled experiments. The dataset maps network-level QoS to application-level QoS [9, 116] and calculates the QoE according to the ITU-T recommendation. For low throughput scenarios, video is downloaded using low resolution but still shows higher QoE for small viewports compared to large ones. Moreover, authors of [10] show that different screen resolutions have different bitrate requirements for the same MOS level. For further information, we recall in Table 3.1 the recommended resolutions for the set of viewports we consider in our experiments [1].

[1] https://support.google.com/youtube/answer/6375112

3.4.1 YouTube chunk resolution pattern

For YouTube, every experiment consists of one video ID and one specific viewport. Once the player is ready, we start the video session and intercept the player's requests for every chunk during playback. Chunk requests include indicators such as video ID, chunk size, and chunk itag that can be used to extract the chunk resolution and codec type. Overall, we stream up to 2K unique YouTube videos using 6 standard viewports. For fairness, we consider only videos available in at least 6 main streaming resolutions (from 144p to 1080p). Our first analysis shows that 99% of the videos streamed were fetched in the video/WebM format, which corresponds to the Google VP9 compression standard. This result confirms the previous observation regarding the Google servers' preference for video/WebM format when serving content.

3.4.1.1 Video resolution pattern

For every chunk request, we leverage the *itag*, *range* and *mime* (Multi-purpose Internet Mail Extensions) parameters to infer the corresponding resolution, the codec and the size. To derive the rate of occurrence of each chunk resolution during the playback of a video on a given viewport j, we use Equation (3.1), where $CR(i,j)$ refers to the set of chunks of resolution i encountered on viewport j. The same formula is used to calculate the video resolution pattern overall videos:

$$ChunkResolutionRate(i,j) = \frac{|CR(i,j)|}{\sum_{i} |CR(i,j)|}. \tag{3.1}$$

The heatmap in Figure 3.4 illustrates the chunk resolution patterns in an easy and interpretative manner. Overall, regardless of the browser viewport, the default start-up chunk resolution is 360p even though lower resolutions matching the viewport are available. Note that the DASH client can still ask for lower resolutions if the network conditions degrade, but it seems that in our case of good network conditions, they are not requested even though some of the viewports we consider require lower resolutions than 360p. Moreover, small viewports such as 240x144, 400x225 and 640x360 form one cluster characterized by the same overall pattern, starting with 360p and scaling up

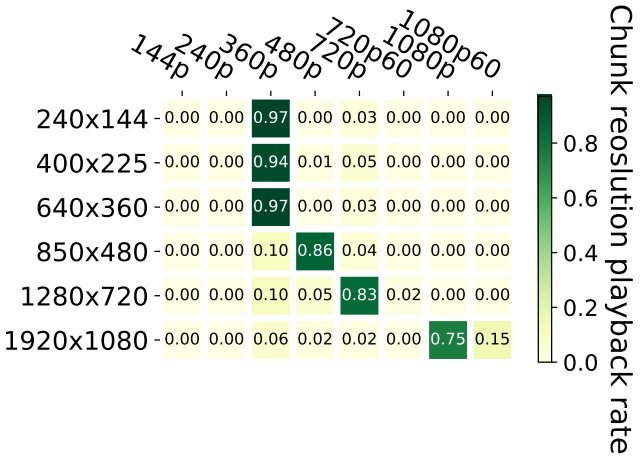
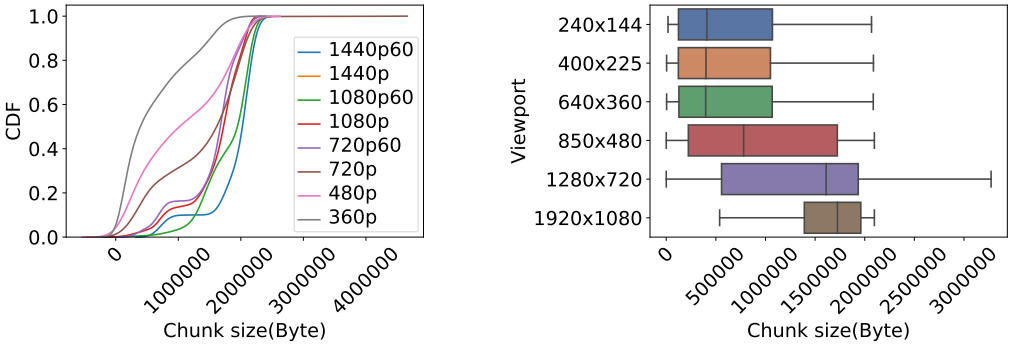


FIGURE 3.4: The YouTube chunk resolution playback rate as extracted from HTTP request clear text

toward higher chunk resolutions (up to 720p). In particular, for 400x225, which is the official YouTube viewport for the small player mode, 94% of the chunks played out are in 360p, only 1% are in 480p, and up to 5% of the chunks are in 720p. We note here that neither the 720p resolution nor the 480p one can be displayed directly in the small player mode. For that, they need to be downsized to match the viewport, thus resulting in what we call bandwidth waste. Meanwhile, the HD and FHD viewports result in chunks of higher frame rates such as 1080p60 and 720p60. As example, the 1920x1080 viewport results in 15% of chunks at 1080p60, which corresponds to a 1080p resolution with 60 frames per second. The normal 1080p resolution from its side is 30 frames per second. We can thus conclude that the first YouTube viewport, which is network friendly, i.e., minimum waste, is the 640x360 one, other smaller viewports exceed the required video resolution.

3.4.1.2 Chunk size analysis

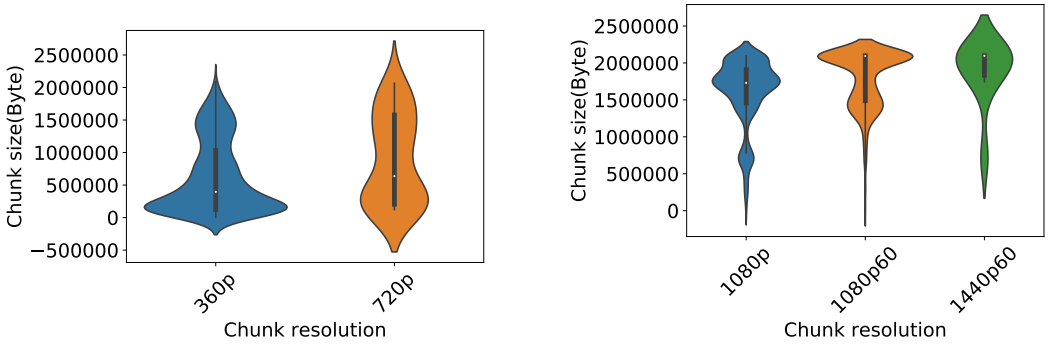
We use the chunk sizes observed in the clear text HTTP traces (from within the browser) to understand the impact of the observed behavior on the bandwidth consumption. We illustrate the results in Figure 3.5. In Figure 3.5(a), we plot the chunk size CDF per resolution. As expected, the chunk size correlates with the chunk resolution, with higher resolutions leading to larger chunks. Here, we make sure to include all encountered chunk resolutions, even those rarely appearing, such as 1440p and 1440p60. Thanks to this



(a) Chunk size CDF per chunk resolution

(b) Chunk size box plot per viewport

FIGURE 3.5: Analysis of YouTube chunk sizes



(a) Chunk size violin plot for 640x360 viewport

(b) Chunk size violin plot for 1920x1080 viewport

FIGURE 3.6: Chunk size violin plots, matching and exceeding resolutions, case of 640x360 and 1920x1080 viewports

information on chunk sizes, we will later estimate the playback bitrate and bandwidth waste given each viewport's observed chunk resolution pattern. In Figure 3.5(b), we plot the distribution of the chunk size per browser viewport. The figure shows that the 240x144, 400x225, and 640x360 screen resolutions form one cluster and exhibit the same chunk size distribution under good network conditions. Larger viewports tend to download larger resolutions. In general, we shall confirm that the pattern of chunk resolutions does carry a signature of the viewport size. In Figure 3.6, we compare the distributions of the sizes of chunks (in Bytes) whose resolution matches or exceeds the resolution of the 640x360 and 1920x1080 viewports (one small and one large). For the 640x360 viewport, the chunks exceeding this display capacity are characterized with a higher median, and 50% of them vary in a larger size space ranging from 150K to 1.5M Bytes. This already gives an idea of the order of bandwidth wasted on these viewports.

3.4.2 Dailymotion video resolution pattern

We perform controlled experiments with up to 200 unique Dailymotion videos on different viewports. To overcome the lack of assets to extract chunk information from raw Dailymotion HTTP messages, we propose an alternative methodology that consists of leveraging the periodic information provided by the Dailymotion API coupled with a probabilistic approach. Instead of intercepting chunk requests as with YouTube, every second our *mainController* receives player updates accessed through its API and stores them in our local database, with every update including (i) the video identifier and title, (ii) the viewport size, (iii) the video duration, (iv) the available video resolutions, (v) the video resolution played out, and (vi) the buffer size occupancy. Later, we use this information to estimate the video resolution playback rate by transforming resolutions into bitrate using statistics on the Dailymotion codec (Figure 3.2(b)). Regarding the list of streaming videos, and as we don't have a public catalog of Dailymotion video metadata, we work with a solution that consists of crawling up to 20 Dailymotion playlists covering several trending categories (e.g., trailers, news, and sports), where every playlist includes on average 10 videos of the same topic. In Figure 3.7, we illustrate the rate of occurrence of video resolutions as reported by Dailymotion for the different considered viewports. These rates are calculated according to Equation (3.2), where $PU(i,j)$ denotes the set of player updates of resolution i on viewport j:

$$VideoResolutionRatio(i,j) = \frac{|PU(i,j)|}{\sum_{i} |PU(i,j)|}. \tag{3.2}$$

As can be noticed in Figure 3.7, the 240x144 viewport only requests the 240p resolution, which is different from what we observed with YouTube. In general, for small viewports, the Dailymotion player does not go high in requesting resolutions (e.g., 720p) even though network conditions can support them. For the 640x360 viewport, Dailymotion shows an interesting behavior, with the 480p resolution being the only one downloaded overall video sessions even though the suitable 360p resolution is well present. One reason could be that the Dailymotion encoder adapts a non-standard width encoding for the 360p, which restricts its use to lower resolution viewports (with a height less than

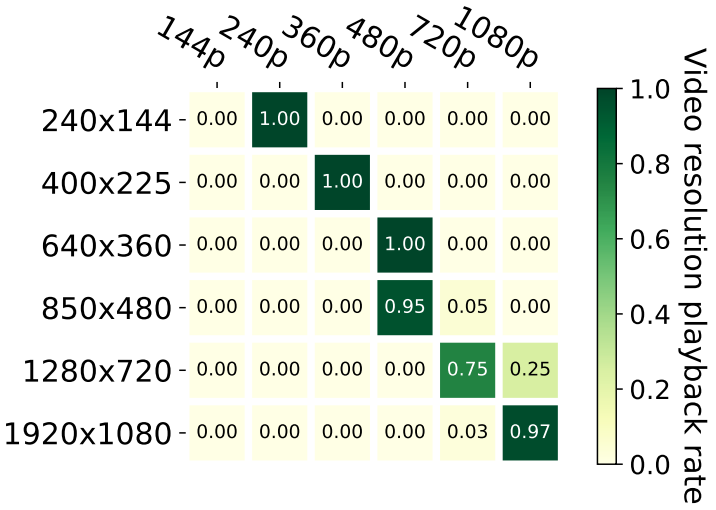


FIGURE 3.7: Video resolution pattern reported by Dailymotion player

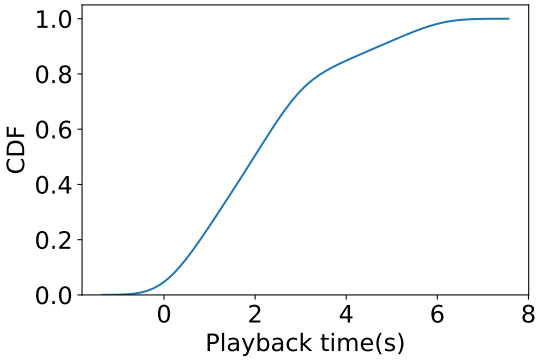


FIGURE 3.8: CDF of 720p video resolution vs playback time for the FHD viewport (1920x1080)

360 pixels). Another interesting behavior is highlighted by the HD viewport (1280x720), where 25% of the reported video resolution updates appear to be 1080p. In other words, for the HD viewport, the Dailymotion player tolerates downloading chunk resolutions exceeding the viewport capacity. These chunks will be unfortunately downsized during playback to match the viewport hence incurring waste of bandwidth. We can thus conclude that for small viewports, the Dailymotion player is more restricted to the viewport capacity than the YouTube player, whereas, for large viewports, it tends to show opposite behavior.

For the 1920x1080 viewport, we can see some low resolutions occurring, as for the 3% of the 720p resolution. We hypothesize that these low resolutions correspond to the start-up phase where the network state is not well estimated. To validate our hypothesis, we

plot the CDF of the 720p video resolution w.r.t. its playback time for the 1920x1080 viewport. Indeed, Figure 3.8 confirms our intuition; all the 720p updates are seen at the very beginning of the video before the 8-th second.

3.5 Quantifying the waste of bandwidth

In this section, we leverage the observed patterns to estimate the bandwidth waste resulting from this behavior. In fact, downloading higher video resolutions than needed does not necessarily contribute to a better QoE as it will be anyways downsized to match the viewport resolution, hence resulting in what we call bandwidth waste. We extend our data-driven analysis to approximate the level of this bandwidth waste.

3.5.1 The estimated playback bitrate

To derive the bandwidth waste for a video, we need its estimated playback rate together with the reference playback rate for the best resolution suitable to the viewport (see Table 3.1). In this subsection, following the results highlighted in Section 3.2, we explain how we estimate the playback bitrate.

3.5.1.1 YouTube playback bitrate

We leverage the experimental results to estimate the real playback bitrate for every video session. For that, we use the chunk size and the video duration. Overall, as in Equation (3.3), the *playback bitrate* is set equal to the sum of chunk sizes of a video session divided by the duration of the session.

$$playback\ bitrate = \frac{\sum_{c \in video\ chunks} size(c)}{video\ duration} \tag{3.3}$$

3.5.1.2 Dailymotion playback bitrate

As explained previously, we cannot infer chunk-related information such as resolution and bitrate from Dailymotion HTTP requests as we did for YouTube. The reason is the lack of open documentation to interpret the raw data embedded in the requests. Instead, we resort to a discrete probabilistic approach that allows us to estimate the playback bitrate of Dailymotion video sessions. Our solution relies on the resolution patterns and the reference Dailymotion bitrate per video resolution. In practice, the *playback bitrate* for a video session is derived according to the following equation:

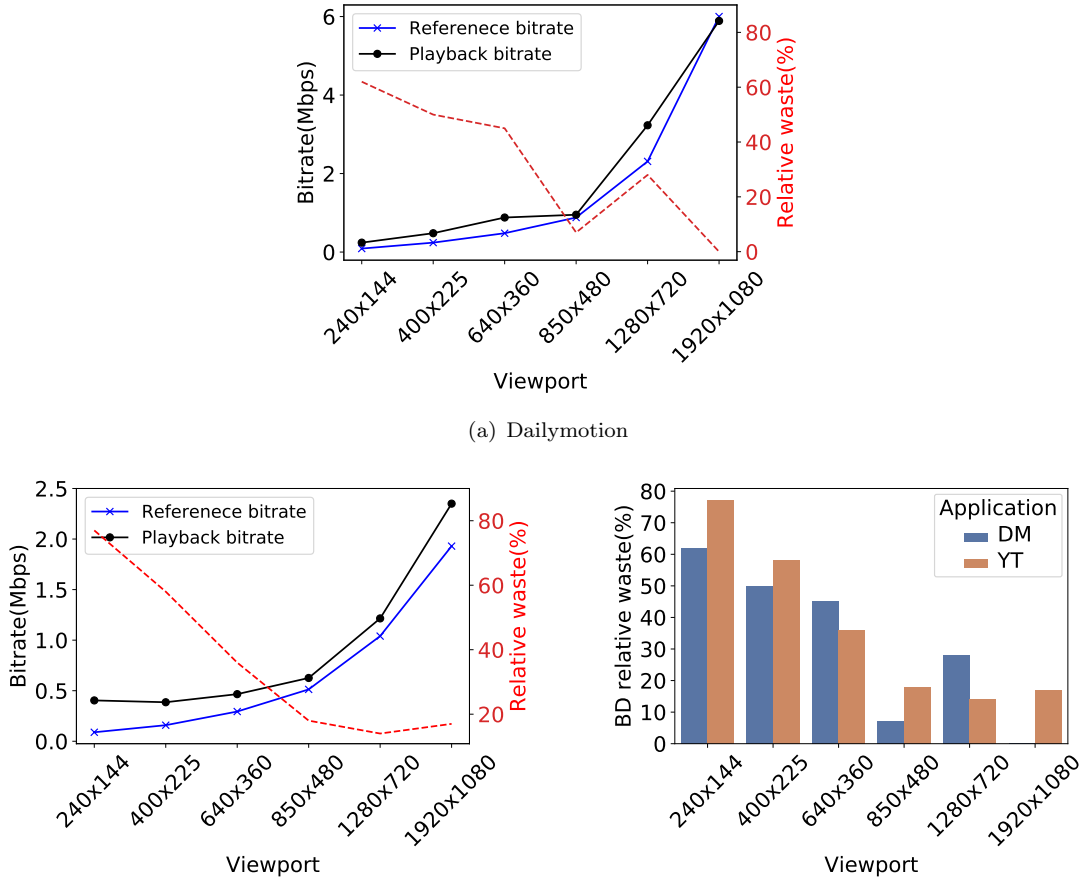
$$playback\ bitrate = \sum_{j \in S} \alpha_j * bitrate_{ref(j)}, \tag{3.4}$$

where S is the set of unique video resolutions after video playback, α_j refers to the video resolution j playback ratio and $bitrate_{ref(j)}$ represents the reference Dailymotion bitrate for video resolution j (as in Figure 3.2(b)).

3.5.2 The estimated bandwidth waste

We define the approximated bandwidth waste for a giving browser viewport as the difference between the estimated *playback bitrate* and its matching *reference bitrate* calculated using the fixed resolution suitable to the viewport as highlighted in Table 3.1.

We plot in Figure 3.9 (red dashed line) the relative bandwidth waste in percent for both YouTube and Dailymotion and different viewports. As a reference, we also plot the average reference and playback rates of both platforms (blue "x" and black "o" lines, respectively). Figure 3.9(a) is for Dailymotion and Figure 3.9(b) is for YouTube. Figure 3.9(c) compares them to each other using a bar plot. A first look at the results shows that no player outperforms the other one for all viewports. Dailymotion comes first for four viewports on six, but this result has to be tempered by the fact that Dailymotion's encoder produces higher bitrates than the YouTube one for the same resolution (see Figure 3.2). We can notice that for small viewports, the Dailymotion player results in less waste on average than YouTube; still, the waste of both players is around 50%. In



(a) Dailymotion

(b) YouTube

(c) YouTube (YT) vs Dailymotion (DM)

FIGURE 3.9: Bandwidth waste

plain, for the small player mode of 400x225, YouTube and Dailymotion players result in 58% and 50% bandwidth waste, respectively. For Dailymotion, the bandwidth waste stays almost at the same level initially, then drops and can reach even 0% over the 1920x1080 viewport. For the 1280x720 viewport, the Dailymotion player reveals a poor performance with 28% waste compared to 14% with YouTube. On the other hand, on the 1920x1080 viewport, the YouTube player shows almost 20% bandwidth waste due to the exceeding resolutions, compared to no waste for Dailymotion.

Overall, the good news comes from the fact that bandwidth waste is negatively correlated to the viewport size. Yet, even for large viewports, the amount of bandwidth wasted cannot be neglected. We project that this waste will have a particular impact on those users with limited data plans. From the side of the network, such debris can improve network efficiency by reducing energy consumption and redistributing the excess bandwidth to other flows in need for it in scenarios of bandwidth shortage.

3.6 Conclusion

To understand the impact of the viewport on the DASH transmission fully, we presented a hybrid methodology combining controlled experiments and a probabilistic approach to investigate the effect of the Chrome browser viewport on the streamed video resolution patterns. To infer the video patterns, we followed two approaches, one for YouTube that consisted of reversing the HTTP clear messages and the other for Dailymotion by using its video player feedback seconded by a probabilistic estimation of playback rate. Later, we reused the discovered patterns to approximate the bandwidth waste for the two video streaming platforms. We also presented a descriptive analysis of an open-source YouTube dataset of more than 1M videos metadata and statistics on the encoders used by YouTube and Dailymotion. Even though the DASH algorithm is supposed to account for the terminal characteristics, our experimental results showed a non-negligible waste of network resources that can be of order 50% for small viewports and 20% for the large ones.

In general, the bandwidth waste tends to decrease when the viewport size increases. However, it remains considerable for operators and end-users to care about. We think in particular about users with limited data plans. We also think about scenarios where concurrent flows can use this excess bitrate, whether videos or other, for a better quality of end-user experience. Reducing the bill of network energy consumption is an objective we have in mind as well.

In the coming chapter, given the importance of the viewport size and the limitations related to end-to-end Internet encryption, we study the correlation of the viewport size with key inband network-level features and provide a methodology to infer them from the encrypted video traces. The aim is to build machine learning models able to predict the viewport size from the encrypted traces.

The contributions related to this chapter appeared in the following publications:

Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On excess bandwidth usage of video streaming: when video resolution mismatches browser viewport**. – *11th IEEE International Conference on Networks of the Future (NOF)* Octobre 2020, Bordeaux, France.

Chapter 4

From encrypted video traces to viewport classification

In the previous chapter, we highlighted through experiments the importance of the viewport resolution in terms of the observed video resolution patterns. Moreover, we could leverage the data produced and quantify the bandwidth waste when the DASH client does not respect the viewport resolution. With that being said, the viewport resolution is an essential factor to be considered for the perceived video experience optimization. Such optimization is of high importance to all stockholders, given the low tolerance level from end-users regarding Internet services. From one side, Internet Service Providers (ISP) engineer their traffic to improve their end-user experience and avoid economic losses. On the other hand, Content providers, and to enforce customers' privacy, have shifted towards end-to-end encryption (e.g., TLS/SSL). For video streaming, and given all the factors to be considered for optimization, the end-to-end encryption makes it a much more challenging task to solve.

This chapter highlights an experimental framework to infer fine-grained video flow information such as chunk sizes from encrypted YouTube video traces. It also features a novel technique to separate video and audio chunks from encrypted traces based on Gaussian Mixture Models (GMM). We leverage our data-driven approach to train models able to predict the class of viewport (either *SD* or *HD*) per video session with an average of

92% accuracy and 85% F1 score. The prediction of the exact viewport resolution is also possible but shows a lower accuracy than the viewport class.

4.1 Introduction

Due to the increasing demand for video content and video services, Internet Service Providers (ISPs) feel more pressure to optimize their networks and meet the expectations of their end-users. They give high importance to video traffic engineering, which requires the ability to infer the context of the video streaming, such as the characteristics of the terminal on which the video is played out and the resolution of the streaming video. However, this is getting more difficult because of the end-to-end encryption adopted by major video streaming platforms (e.g., YouTube, Netflix, and Amazon) [29]. For example, to prioritize or load balance video traffic efficiently, ISPs need information on end-users ' Quality of Experience (QoE) rather than just capturing the network Quality of Service (QoS). But, video QoE is dependent on the content itself (the video bitrate and resolution) and on the application-level QoS metrics such as start-up delay, duration of stalls and resolution switches [16, 111, 117]. It also depends on the resolution of the viewport on which the video is played out [10]. Unfortunately, all this information is hard to obtain from encrypted video traffic, making its inference an important challenge to surmount.

This chapter presents a data-driven methodology unveiling the end-user viewport resolution from YouTube encrypted video traces. To that aim, we leverage video chunk sizes and inband network-level traffic features such as throughput and download/upload packet inter-arrival times to train machine learning models able to distinguish between *HD* and *SD* viewports and infer the resolution of the viewport. More specifically, our contributions are the following:

- We present a controlled experimental framework to perform video streaming experiments large-scale and collect YouTube video metadata. We leverage the Chrome Web Request API to read the clear HTTP text messages [5] and obtain ground truth on the video streams and the dynamics of their chunks.

- We stream up to 5000 unique YouTube videos, collect encrypted traces and clear HTTP messages, and show that chunk sizes and inband network-level traffic features carry an interesting signature of the viewport resolution.
- We propose a novel approach to separate video and audio chunks from encrypted video traces based on a Gaussian Mixture Model (GMM). Then, we validate our work on inferring video chunk sizes by comparing similarities and differences concerning the real video chunk size distribution derived from clear HTTP messages.
- We train different machine learning algorithms to classify the viewport resolution. We prove the pertinence of this classification, taking as input video chunk statistics and inband network-level traffic features that can be derived from passive captures of encrypted video traffic.

Overall, this chapter is organized as follows. In Section 4.2, we present our experimental framework, followed by a descriptive study of today's video content based on an open-source YouTube catalog. In Section 4.3, we present our methodology to extract video chunk sizes from YouTube encrypted traces. Later, in Sections 4.4 and 4.5, we highlight the viewport signature carried by a set of inband network traffic features and chunk size statistics and train and evaluate a classifier able to classify the viewport resolution from encrypted traces. Last, we conclude our work in section 4.6.

4.2 Experimental setup

We play different YouTube videos using different viewports and under various network conditions emulated using Linux traffic control (*tc*). Each experiment consists of a unique YouTube video, browser viewport, and enforced network bandwidth. We leverage the Chrome Web Request API for every video session to read the clear HTTP text messages and establish ground truth on the requested chunks and the application-level quality of service. Moreover, we dump the encrypted client-server traffic to pcap files using *tcpdump*.

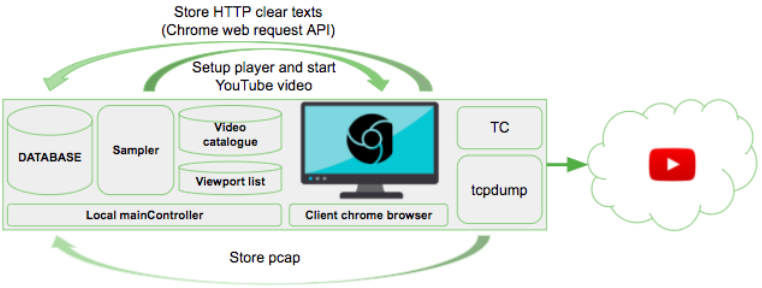


FIGURE 4.1: Experimental framework description

4.2.1 Overall experimental framework

Our overall experimental setup, described in Figure 4.1, is almost the same as what we used for the bandwidth waste study (Section 3) with key differences related to network emulation and traffic capture. To summarize again it consists of; (i) a local *mainController* (MacBook Air machine of 8 GB RAM. Videos are visualized on a Dell screen 27' of 2560 x 1440 resolution, a (ii) YouTube video catalog, and (iii) the viewport list as illustrated in Figure 4.1. We consider a list of default standard viewports representing current player dimensions adopted by streaming platforms for several watching modes. Moreover, in this chapter, and since the video resolution pattern is a function of both network conditions and terminal display capacity, we study the viewport importance while degrading the network bandwidth. To that aim, we use Linux traffic control *tc* and enforce different bandwidth settings such as 3, 6, 9, 15, and 20 Mbps. We also stream with no bandwidth limitation on Ethernet to emulate the best-case scenario. Moreover, we use *tcpdump* tool to dump the client-server traffic into *pcap* files.

4.3 Analysis of video streaming traffic

In adaptive video streaming, the client decides on the resolution of the next chunk to download based on underlying network conditions and viewport characteristics. So each viewport is supposed to exhibit a different video resolution pattern depending on the available network resources; the pattern of chunk sizes is the main illustration of such specific behavior. However, as most of the video traffic is encrypted, the information on the viewport resolution is not visible to any entity between the client and the server.

Our intuition is to exploit the specificity of the chunk size pattern, and other **in-band** network features to infer the viewport resolution from encrypted traffic. The problem is that when the bandwidth starts getting scarce either due to congestion or to in-network shaping, clients are automatically forced by DASH to request lower video resolutions, thus reducing the effect of the viewport and increasing the difficulty to infer its resolution. To highlight these aspects, we investigate the extent to which screens impact the video transmission pattern while varying the network bandwidth.

4.3.1 Inferring video chunk sizes

Overall, we stream up to 5K YouTube unique videos in series randomly selected from the 1 Million catalog in [104]. In general, chunks of a video are fetched using separate HTTP requests. On the one hand, we infer the chunk sizes from the encrypted YouTube traces. In parallel, we extract the real chunk sizes from the clear text HTTP messages accessed from within the Chrome browser using the Chrome Web Request API [5]. Our chunk size inference method is inspired by [9, 82] and works as follows. At first, we use the source IP of our host and the list of destination IPs to isolate the different flows corresponding to every video session (CDNs identified by the URLs ending with googlevideo.com). The CDN identifiers can be collected from the clear HTTP text messages, and their corresponding IP addresses can be resolved. Then, for each video streaming session (source and destination already known), we look at the size of uplink packets. The large size uplink packets correspond to chunk requests, while small packets correspond to transport-level acknowledgments by TCP/QUIC. Instead of using thresholds as depicted in [82], we use K-means clustering to segregate the uplink packet sizes into two clusters; the first cluster represents the request packets, and the second cluster represents the acknowledgment packets. Once the uplink request packets are identified, we sum up the data downloaded between any two consecutive request packets and consider it equal to the downloaded chunk size following the first request between the two. Overall, we leverage clustering for the sake of generality and to make sure our approach can be reused with other types of ACKs, mainly in the context of different transport protocols.

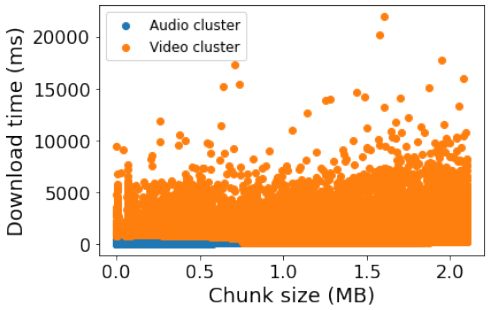


FIGURE 4.2: Audio/video clusters as produced by GMM

Up to this point, and as the case for other existing methodologies, the calculated chunk sizes mix between audio and video chunks, whereas we are only interested in the video part. The chunks located between consecutive request packets can be either of the two types, audio or video, and so need to be separated. To overcome this limitation, we leverage Gaussian Mixture Models (GMM) applied to the chunk size. The GMM clustering method is based on the maximum likelihood principle, finding clusters of points in a dataset that share some common characteristics. Unlike K-means, the GMM belongs to the soft clustering subset of unsupervised algorithms. It provides probabilities that tell how much a data point is associated with a specific cluster. Another critical property of GMM is that clusters do not need to be topologically separated as with K-means. They can overlap and still be identified as long as they follow some Gaussian property for the distribution of their points. In general, video chunks should have larger sizes than audio chunks, and we rely on this property to identify the two Gaussian distributions and classify the chunks between audio and video. Each distribution has three unique values, mean γ, covariance $\sum$ modeling the spread around the mean, and a probability π defining how big or small one cluster is compared to the other one, the sum of probabilities of the two clusters is naturally equal to 1.

For our case, we fit a GMM of two components with the chunk sizes inferred according to K-means. For a clear visual illustration, we plot in Figure 4.2 the two clusters rendered by the GMM method over a 2D space of chunk sizes (MB) and download time (s). In plain, the audio cluster (in blue) shows chunks smaller than 750 KBytes with no more than 2 seconds of download time. Video chunks (in orange) can be of larger sizes and download times compared to audio ones.

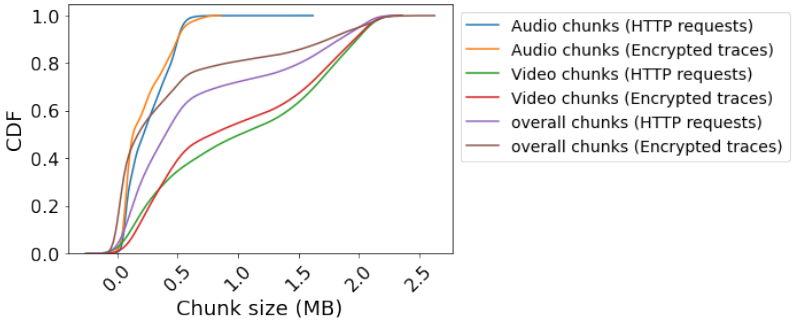


FIGURE 4.3: Chunk size CDF

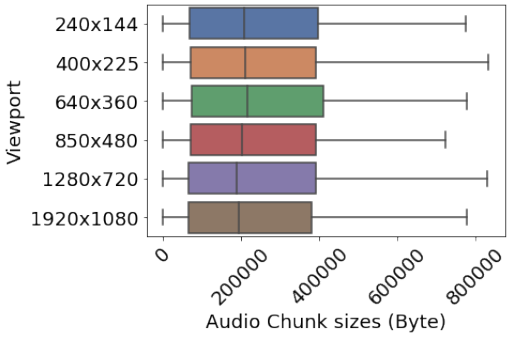


FIGURE 4.4: Audio bitrate distribution

Now we test the accuracy of our method by comparing its output to the ground truth collected directly from within the browser by analyzing the explicit HTTP requests. These requests include the *itag*, *range* and *mime* (Multi-purpose Internet Mail Extensions) parameters, which can be then used to infer the corresponding resolution, the codec, and the size of the chunk using open-source documentation [113]. We use this ground truth to check whether our GMM method provides video chunk sizes that respect the distribution of the size of real video chunks as seen in the browser. Figure 4.3 compares the chunk sizes as estimated by our method from the encrypted traffic traces and the chunk sizes obtained from the clear text HTTP traces for the same video sessions. The overall distribution of the encrypted chunk sizes extracted using our method exhibits the same shape as those obtained with HTTP requests. Further, the two distributions produced by our method for audio and video chunks are very close to those of the HTTP requests. We can also notice how the video chunks, understandably, have larger sizes than the audio chunks. This helps better characterizing the video chunks within a trace of encrypted video traffic, with this result particularly useful in our case to understand further the interplay between viewport, network resources, and chunk resolution pattern.

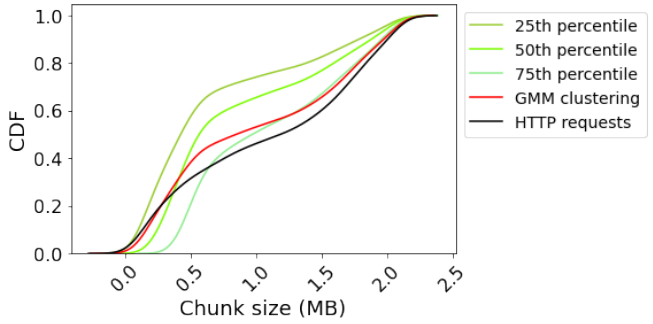


FIGURE 4.5: Threshold/GMM/HTTP inference of video chunks

4.3.2 Audio chunk size distribution

We illustrate in Figure 4.4 the audio chunk sizes accessed from within the Chrome browser using the Chrome Web Request API [5] w.r.t. the viewport resolution considered in the experiments. Overall, we notice that regardless of the viewport resolution, the audio chunk size distribution is almost the same, which discards any impact of the viewport and confirms the use of standard audio quality. In plain, the audio chunk size distribution is characterized by a median encoding bitrate of 200 Kbytes. Moreover, the audio chunk size variation is almost the same through all viewports, with the 25th percentile and 75th percentile equal to 100 and 400 Kbytes, respectively.

4.3.3 Threshold based audio/video chunk separation

Above, we leveraged the GMM clustering to separate audio and video chunks from each other. Another feasible solution easier to deploy would be to use static thresholds applied to chunk size. Here, we compare clustering and threshold-based techniques for the sake of chunk segregation efficiency.

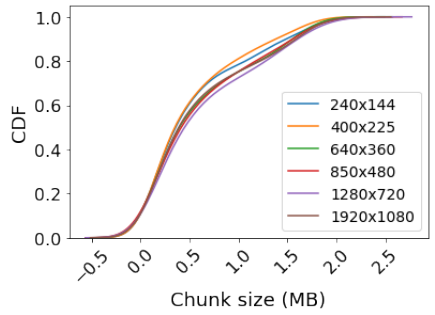
We leverage the audio chunk size distribution illustrated in Figure 4.4 to derive threshold values able to separate the two types of chunks based on their sizes. We use three threshold values representing the 25th, 50th, and 75th percentiles of audio chunk sizes in plain. For each threshold, the set of audio chunks include every chunk with a size less than the threshold, while the others are considered video chunks. In Figure 4.5, we plot the video chunk size distribution per different separation methods; *(i)* the three variants of the threshold-based method, *(ii)* the video chunk sizes inferred using the

GMM clustering, *(iii)* and the real video chunk size distribution as inferred from the HTTP requests. We can observe that the overall distribution of the video chunk sizes is well captured by both the threshold-based and the clustering-based methods, with as expected, the higher the threshold, the more the shift of the distribution towards larger video chunks. In plain, the 75th percentile threshold provides the closest distribution to the real one, yet the GMM method using the maximum likelihood principle can capture the real video chunk size distribution in a close manner. To note here that a main advantage of the GMM clustering method is in its automatic learning property, which prevents one from tuning the threshold value manually.

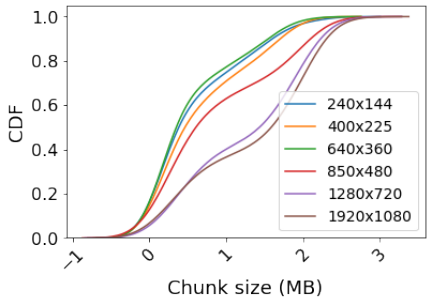
4.3.4 Video resolution pattern

The DASH client automatically switches between video resolutions according to the viewport and underlying network performance. The video resolution pattern as requested from the server is thus determined by the network conditions and normally has to take into consideration the viewport resolution, which is defined as the number of pixels, both vertically and horizontally, on which the video is displayed. It is indisputable that the network conditions, for instance, the bandwidth, reduce the screen's impact in scenarios of bandwidth shortage as DASH will download chunks of lower resolution than the viewport capacity. In this section, we present experimental results supporting these statements and highlight in particular, the reduction of the effect of the viewport as the available bandwidth decreases.

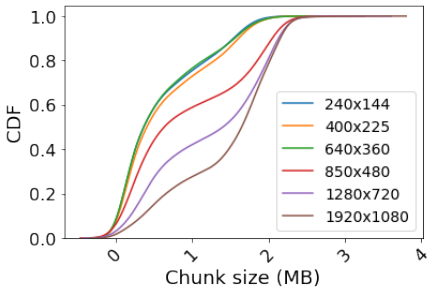
We artificially change the available bandwidth (as highlighted in Section 4.2), and stream for each bandwidth setting hundreds of YouTube videos using different viewports. Each time, we use random sampling to select the video ID and the viewport. We plot in Figure 4.6 the CDF of the video chunk size per viewport for three bandwidth settings: 3 Mbps, 15 Mbps, and no control. As expected, the video resolution pattern is driven by the network bandwidth and the viewport resolution. In Figure 4.6(a), the bandwidth is limited to 3 Mbps, and all viewports thus exhibit the same pattern by streaming the same video resolution, which therefore results in the same cumulative distribution of chunk sizes. However, in Figure 4.6(b), we set the bandwidth to 15Mbps, the effect of



(a) Chunk size per viewport with a 3Mbps bandwidth



(b) Chunk size per viewport with a 15Mbps bandwidth



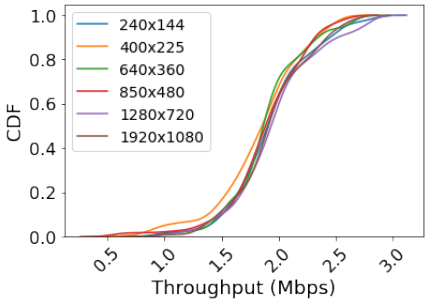
(c) Chunk sizes per viewport with unlimited bandwidth

FIGURE 4.6: Network and viewport impact on chunk sizes

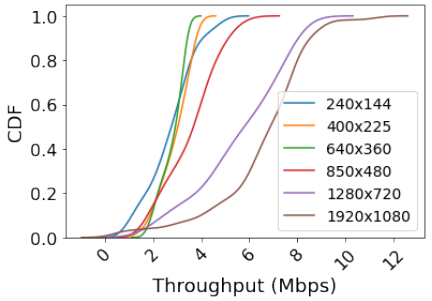
the viewport starts appearing as the distribution of chunk sizes differs from one screen to another. However, and even at this high bandwidth, the two large viewports 1280x720 and 1920x1080 illustrate close distributions, which can be explained by the same reason of bandwidth shortage. Finally, when no restriction is imposed on the bandwidth, a high-definition viewport (1920x1080) starts differentiating itself from the others. We further notice in Figure 4.6(c) that 40% of chunk requests on small screens (e.g., 240x144, 400x225 and 640x360) correspond to a chunk-size smaller than 200 KBytes compared to 300 KBytes for medium screens. For 1280x720 viewports (HD), chunk sizes are bigger, with 40% of them smaller than 1 MBytes (resp. smaller than 1.6 MBytes for 1920x1080 Full HD viewports).

4.4 Traffic correlation to viewport

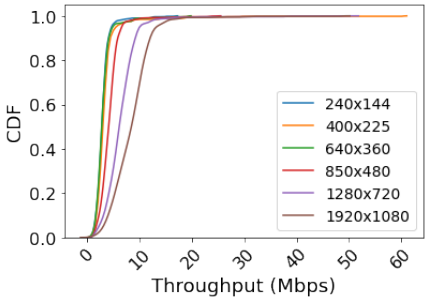
To illustrate this result further, we plot the network throughput as measured over the encrypted traces and compare it to the available bandwidth for different viewports. For each video, we get the CDN URL from the HTTP logs and use the DNS Lookup of



(a) Throughput per viewport for a 3Mbps bandwidth



(b) Throughput per viewport for a 15Mbps bandwidth



(c) Throughput per viewport for a unlimited bandwidth

FIGURE 4.7: Throughput per viewport for multiple network settings

the CDN URL to identify the video flow corresponding to using the CDN IP. Then, we leverage the downlink packet timestamps and a time bin of 1s to return a vector of throughput values per video session. The vector is used to derive throughput statistics (e.g., average, percentiles) per video session. In Figure 4.7, we plot the CDF of the throughput values of the video sessions for different viewports with different bandwidth settings. We notice that with an enforced bandwidth of 3 Mbps (Figure 4.7(a)), all viewports end up experiencing the same throughput, which correlates with chunk size results. Moreover, regardless of the available bandwidth, a subset of viewports form one cluster exhibiting the same throughput pattern (e.g., 240x144, 400x225, and 640x360).

For each video session, we get an array of video chunk sizes over which we calculate different statistical features that we plan to use for viewport classification. We study here the correlation between this array and the viewport. Our feature set contains the maximum, the average, and the standard deviation along with the $10th$ to $90th$ percentiles (in steps of 10) of the chunk size array. This forms a set of 12 features describing the evolution of chunk sizes over a video session statistically. In addition to chunk size-related

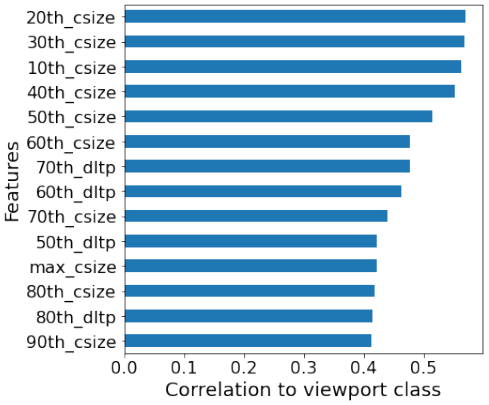


FIGURE 4.8: Features correlation to viewport class

statistics, we also consider the same statistical features, but this time for the downlink throughput (in bps, averaged over time bins of 1s) and the uplink and downlink packet interarrival times (in seconds). We believe that we get a fine-grained description of the DASH transmission process and capture any effect of viewport resolution with these features. Overall, according to feature analysis, viewports such as 240x144, 640x360, and 850x480 are more likely to exhibit close chunk size and throughput distributions forming one viewport class (*SD*). On the other hand, the 1280x720 and 1920x1080 represent another cluster, called *HD* showing similar properties. To take advantage of this overlapping, we equally consider a relaxed definition of the viewport classification problem to either *SD* or *HD*.

Before building our classifier, we illustrate the correlation between our feature set and the viewport class. Figure 4.8 points to the most relevant features by ranking them according to their Pearson correlation coefficient with the viewport class. The figure shows only those features having a correlation coefficient at least equal to 0.4. The x_th_csize represents the x_th percentile of the video chunk size over a video session and the y_th_dltp stands for the y_th percentile of the downlink throughput. Overall, the chunk size percentiles show a more critical correlation with the viewport capacity, especially when it comes to lower percentiles. This is because the video resolution pattern is not only influenced by the available network resources but also by the user display capacity. Downlink throughput percentiles come in second place with a correlation coefficient of more than 0.4.

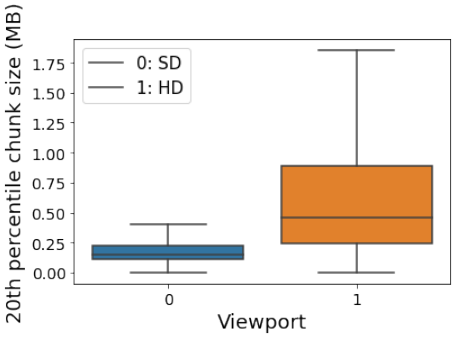


FIGURE 4.9: $20th$ chunk size percentile (most relevant chunk size percentile in Figure 4.8), *SD* (0), *HD* (1)

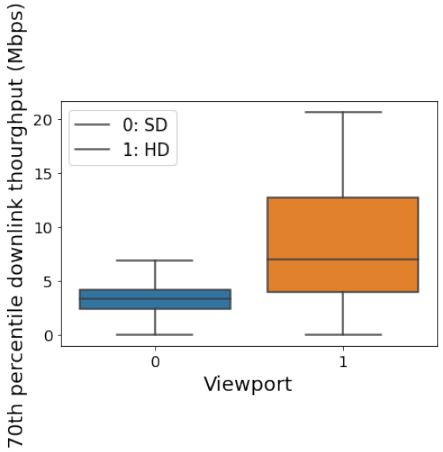


FIGURE 4.10: $70th$ downlink throughput percentile (most relevant throughput percentile in Figure 4.8), *SD* (0), *HD* (1)

To shed further light on the previous results, we show boxplots of the essential traffic features w.r.t. the two viewport classes. We plot in Figure 4.9 the distribution of the $20th$ chunk size percentile for all video sessions and both viewport classes. Overall, we notice a small overlapping portion; the smaller the overlap, the easier it to differentiate between *SD* and *HD* viewports. In plain, 50% of video sessions have their $20th$ chunk size percentile less than or equal to 230 KBytes, whereas high definition viewports score almost twice the value for the same percentile. In terms of downlink throughput, we plot in Figure 4.10 the distribution of the $70th$ download throughput percentile for all video sessions as it scores 0.46 in terms of the correlation coefficient. In general, and as expected, larger screens are characterized by larger throughput values. Moreover, the boxplots show that half of our video sessions have a $70th$ download throughput percentile around 7 Mbps compared to 4 Mbps for small definition viewports. All these results point to a correlation pattern between encrypted traffic features and viewport

resolution at the client, a pattern that we will exploit next to build our classifier of viewport resolution.

4.5 Viewport classification by machine learning

In this section, we discuss the performance of the ML model built using our dataset. We start by predicting the viewport class (*SD* or *HD*) using inband network features and chunk size stats ($F_{inband+chunk}$) extracted from the YouTube encrypted traces. Later, we highlight the performance of our methodology in the context of multi-label classification, where the viewport resolution is precisely targeted. Our goal is to provide the ISP with a means to infer viewport resolution insights despite the end-to-end encryption of the video flows. Such inference can help the ISP get an idea about the bandwidth requirements of their customers and their level of Quality of Experience (QoE) with the obtained network service. The latter can enhance network management decisions (e.g., resource allocation priority queuing) to improve such QoE.

We build a dataset matching $F_{inband+chunk}$ to viewport capacity and use it to train different supervised ML classification algorithms. We randomly pick videos from the catalog available in [104], then stream them under different network conditions emulated locally using the Linux *tc* utility. Each experiment consists of enforcing the bandwidth, playing out the selected video under the enforced QoS, collecting clear HTTP messages using the Chrome Web Request API, and dumping the traffic in *pcap* files using tcpdump. The *pcap* files are used to calculate the feature set $F_{inband+chunk}$.

4.5.1 Viewport class classification

As we have seen before, the effect of the viewport is maximum in an unlimited bandwidth scenario. As bandwidth decreases, the different viewports converge to the same video resolution pattern. Therefore we expect any viewport inference model to become less accurate as both classes (SD/HD) start overlapping. To assess the extent of such limitation, we test our model in different scenarios, each featuring a different bandwidth configuration. We use Random Forest (available in python Scikit-Learn library [118])

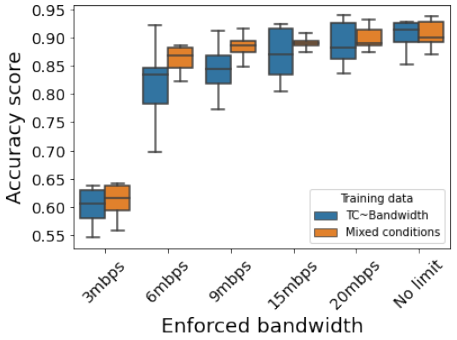


FIGURE 4.11: Model accuracy vs enforced bandwidth

because of its out-performance in our case compared to other classifiers such as Support Vector Machine, Decision Tree, and Multi-Layer Perceptron. To find the best tuning of the Random Forest algorithm, we apply at first a random search of the best hyper-parameters values. Then after reducing the search space, we use a grid search to get a fine-grained fitting of major parameters [119].

Figure 4.11 highlights the accuracy of two classification models trained with our dataset. In plain, for each bandwidth setting on the x-axis, we highlight two Random Forest models trained on two different datasets; the blue model is trained with video samples conducted with one specific enforced bandwidth (the corresponding x-axis value), and the orange model trained with the aggregate set of video sessions obtained overall executed bandwidth values. The blue model varies from one x-axis value to another one, whereas the orange model is the same overall x-axis values. We validate both models on a test set of 200 videos specific to each bandwidth scenario. In general, regardless of the training set, the model accuracy is coherent with our intuition and increases w.r.t. the enforced bandwidth. For example, in the case of enforced bandwidth of 3Mbps, both models show a low performance with a median accuracy of 62%. This is expected as for such low bandwidth, viewports show similar distributions for most important features (see Figure 4.6). One can expect an even lower accuracy if the exact viewport resolution is predicted for such a low bandwidth value. Starting from 6 Mbps, both models show a median accuracy exceeding 80%, with the model based on mixed conditions showing better accuracy (in terms of both average and variance) than the model specific to the enforced bandwidth value, which is a good property of the orange model given its generality over different bandwidth scenarios. We recall that the best performance for both

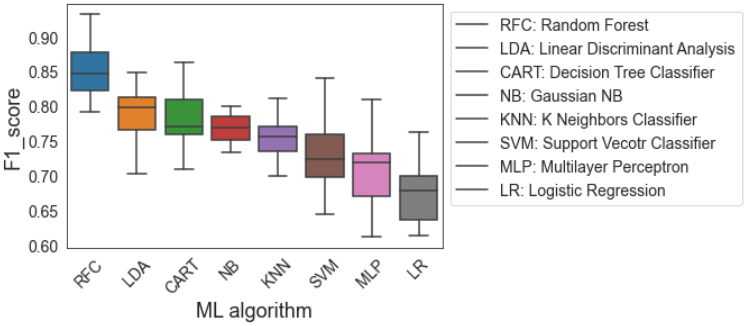


FIGURE 4.12: ML algorithm comparison (no bandwidth limitation)

models is reached when no limitation is imposed on the network bandwidth with 92% median accuracy.

The previous results present a general evaluation of the model, yet, we need to evaluate the model per viewport class. Here, one can use metrics like *precision* and *recall* or simply the F1 score, which is an average of both. To that aim, we benchmark a set of well-known supervised machine learning algorithms, fit them on our dataset (for the no bandwidth limitation case) and compare them per class using the F1 score. We plot in Figure 4.12 the k-fold (k =10) validation results for the set of machine learning algorithms we consider. According to this validation, the dataset is split into k folds, and at each of the k iterations, a new fold is used as a validation set while the k - 1 remaining folds form the training set. Average results are then calculated over the k iterations. In plain, Random Forest seems to be the most relevant algorithm, as it shows an average F1 score of 85%, while the other classification algorithms such as Linear Discriminant Analysis and Decision Tree come second and third with average F1 scores of 80% and 78% respectively. The lowest performance is recorded for the Multi-Layer Perceptron and Linear Regression classifiers with 72% and 68% F1 scores, respectively. Moreover, we show in Table 4.1 the precision and recall values of a tree sample produced by our Random Forest model. The model classifies correctly 85% of the total video sessions issued from *HD* viewports and 93% of the sessions related to *SD* viewports. When our model labels a video session as *HD*, it is correct in 87% of the cases and 89% of the cases for *SD*.

	Precision	Recall	F1
HD	0.87	0.85	0.86
SD	0.89	0.93	0.91

TABLE 4.1: Random Forest case (precision/recall)

4.5.2 Viewport resolution classification

The previous analysis highlights the performance of our model in a binary scenario of *SD/HD* viewport classes. In this subsection, we illustrate the performance of our model in a multiclass scenario, where we aim at predicting the exact viewport resolution as used in the experimental setup (see Figure 4.1). We show the Random Forest model results trained with video sessions conducted in the no bandwidth limitation scenario. We leverage a heatmap to highlight the prediction accuracy of our model per viewport resolution.

We plot in Figure 4.13 the confusion matrix of the predicted viewport resolutions. The y-axis (rows) corresponds to the ground truth on viewport resolutions, while the x-axis (columns) represents the predicted ones. The value in case (i, j) represents the percentage of viewports of size i that are classified as of size j, the sum of elements in a row is equal to 100%. The color intensity of a case increases with its value. According to this heatmap, one can identify two regions, (i) small viewports mainly from 240x144 up to 850x480, and (ii) high definition viewports of sizes 1280x720 and 1920x1080. The heatmap shows the difficulty of classifying video sessions on small viewports. For instance, 44% and 37% of the video sessions on 400x240 and 640x360 viewports are labeled as 240x144. On the other hand, for large viewports, most video sessions can be classified correctly with our set of features, as 54% and 60% of video sessions on 1280x720 and 1920x1080 viewports are correctly labeled. In the middle, the 850x480 viewport is the one with the largest uncertainty in the classification between the two viewport classes, with 28% of its video sessions labeled with low viewport resolutions and 12% with superior viewport resolutions. It is for this reason that we decided to consider viewports of this size as belonging to the *SD* class.

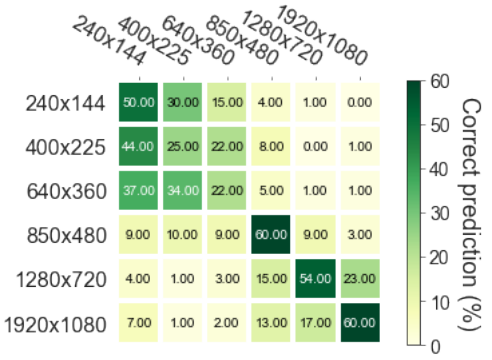


FIGURE 4.13: Viewport resolution classification

	Precision	Recall	F1
240x144	0.32	0.52	0.40
400x225	0.26	0.24	0.25
640x360	0.31	0.24	0.27
850x480	0.52	0.62	0.57
1280x720	0.59	0.52	0.55
1920x1080	0.70	0.54	0.61

TABLE 4.2: Multi-class case: Precision/Recall & F1-score

Following the same analysis for binary classification, we highlight in Table 4.2 the classification performance per class using the Precision/Recall and F1-score metrics. In plain, for the 400x225 and 640x360 viewports, the F1-score is the lowest with 25% and 27% respectively, hence highlighting the model's confusion when it comes to video sessions on small screens. The model performs better with an F1-score of 55% for the 1280x720 viewport and 61% for the 1920x1080. This relatively low accuracy of the classification in the multiclass scenario is expected, as our analysis has pointed to two different subsets of viewports (*SD* and *HD*) presenting close properties internally for their inband network features and chunk size statistics. The result is a space of collision inside each subset and confusion regions where the model is highly uncertain. Luckily the overlap is less significant between the subsets leading to the good performance of the binary classification case.

4.5.3 Real-time viewport classification

The statistics we used so far to train and test our model consider the entire video session. This requires waiting until the end of the session to collect the features and predict the viewport, limiting the usability of the method in practice by preventing from taking real-time traffic engineering actions. One needs to perform the classification as soon as the video starts playing out, thus allowing for mechanisms such as weighted fair queuing and load balancing to take place. So here we study the goodness of our model for viewport classification on the fly, which instead of using as input aggregated statistics on the entire video session, calculates features on the early part of the session.

We stream a total of 104 hours (4 days) and 70 hours (almost 3 days) of random YouTube videos using different *SD* and *HD* viewports, respectively. We highlight in Figure 4.14 the video duration distribution per viewport class. As expected, the two distributions look the same, with half of the videos requested from *SD* and *HD* viewports having a median duration of 120 seconds. We split this dataset into training and test sets. In the training set, we compute the features by considering the entire video session. On the other hand, we use a specific proportion of the video starting from its beginning and test over it for the test set. For instance, an input on the first 20% will consider a feature set $F_{inband+chunk}$ calculated over the first 20% of the video, and so on. To represent the proportions in seconds and give them a practical meaning, we consider the median video duration (120s) as reference duration, so proportions of 20% and 40% would correspond to the first 24 and 48 seconds of a video session.

We plot in Figure 4.15 the F1 score w.r.t. the proportion used as input for the test. With no surprise, the more significant the considered proportion of video sessions is, the higher the accuracy of the model becomes. This makes sense as the model gets more relevant input than those used for the training. More importantly, the model still works with few seconds as input and provides good classification accuracy exceeding 80% on average. This confirms that the first few seconds of a video session carry an important signature of viewport class, for example, the first 24 seconds (assuming a median duration of 120 seconds), allowing a median F1 score of 80% (more than 78% in 75% of the cases). We recall that considering the complete video data leads to an 85%

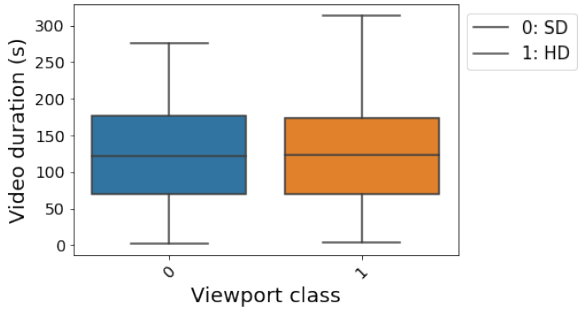


FIGURE 4.14: Video duration distribution (seconds)

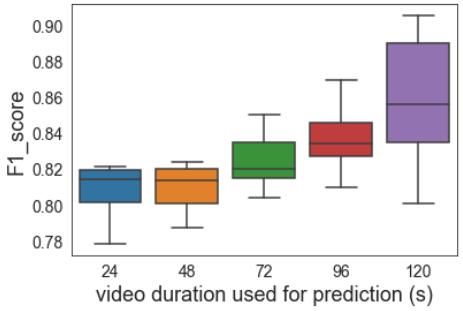


FIGURE 4.15: Model accuracy vs video proportion considered

median F1 score. We shall thus confirm the feasibility of our approach for pseudo-real-time viewport classification.

4.6 Conclusion

In this chapter, we presented our methodology for building viewport classification models from YouTube encrypted video traces using controlled experimentation and machine learning. Our models infer the end-user viewport resolution from statistical features calculated over the encrypted video packets, fully or partially. Such information on the viewport can help the ISPs plan better traffic engineering actions for a more efficient network management and QoE optimization. Our methodology starts by inferring chunk sizes, then relies on Gaussian Mixture Models (GMM) to separate video chunks from audio chunks. Statistics on video chunks are then used to train machine learning models for viewport classification. In a binary scenario of *SD* and *HD* viewports, our models showed classification accuracy that improves with the available network bandwidth and can go up to 92% in its median. The median F1 score can go up to 85%. Limiting the classification to the first few seconds of the video decreases its accuracy but still leads

to acceptable levels of F1 score. The inference of the exact viewport resolution showed a lower accuracy, as a subset of viewports presents similar statistical features, making the prediction more challenging to realize.

In the rest of the thesis, we will explain how to bridge the gap between video QoE optimization and viewport resolution. For that, we will study a resource allocation problem using a QoE function that considers the viewport resolution along with other metrics to allocate bandwidth and cache content intelligently.

The contributions related to this chapter appeared in the following publications:

- Othmane Belmoukadam and Chadi Barakat: **From encrypted video traces to viewport classification**. – *16th International Conference on Network and Service Management (CNSM)*, November 2020, Virtual Conference - BEST PAPER AWARD
- Othmane Belmoukadam and Chadi Barakat: **From encrypted video traces to viewport classification**. – *The World of Industrial Mathematics MOMI2021*, March 2021, Virtual Workshop - BEST POSTER AWARD
- Othmane Belmoukadam and Chadi Barakat: **Unveiling the end-user viewport resolution from encrypted video traces**. – *IEEE Transactions on Network and Service Management*, May 2021.

Chapter 5

QoE-aware bandwidth sharing framework for adaptive video streaming

Up to this point, we were able to enlighten the impact of the viewport resolution on the video resolution patterns and data consumption. We highlighted the correlation of the viewport resolution to key inband network-level features (e.g., chunk size) and proposed a data-driven solution to infer this viewport from encrypted traffic traces. At this stage, we reuse this information to bridge the gap with video QoE models. Furthermore, we incorporate our models to solve problems related to Internet resource management. In a nutshell, in this chapter, we define a QoE-aware resource allocation problem to pinpoint the optimal bandwidth allocation that maximizes the QoE overall users of a network service provider located behind the same bottleneck link while accounting for the characteristics of the screens they use for video playout. For validation, we use ns-3 and show that our solution can increase the overall QoE compared to an allocation with a TCP look-alike strategy.

5.1 Introduction

In light of video traffic growth and equipment diversity, video traffic engineering is a very challenging task to solve. The difficulty stems in particular from the different requirements of viewports from the side of the network. Previous studies linked the MOS (Mean Opinion Score) to the video bitrate for different screen types (e.g., Common Intermediate Format (CIF), Quarter Common Intermediate Format (QCIF), and High Definition (HD)). They show that small screens scale faster to higher MOS levels even when displaying videos of limited resolution, suggesting different media-related requirements for the same QoE level [10].

Generally speaking, for QoE models insinuating the IQX hypothesis [28], i.e., exponential mapping between QoE and QoS metrics, the QoE variation resulting from a change of a QoS metric depends on the current QoE level. Within this hypothesis, if displaying a low video resolution on a small definition screen results in a good QoE level, a slight degradation of the QoS (e.g., bandwidth) will not impact the perceived QoE drastically. However, displaying the exact low video resolution on a large definition screen will result in a low QoE level so that the same QoS degradation will have a more severe impact on the QoE. This justifies the need for differentiated treatment of screen resolutions inside the network. Unfortunately, this differentiation is challenging to in-store for several reasons; (i) despite the importance of the viewport resolution, little is known when it comes to incorporate this important variable, when allocating network resources (e.g., bandwidth) and (ii) the allocation is often left to the HAS and TCP protocols, which in case of multiple flows sharing a bottleneck link, converges to a fair split of the available bandwidth. For different screen resolutions or different viewports in general, this latter allocation does not lead to a fair QoE allocation nor an optimal overall QoE.

To that aim, motivated by the previous results and previous studies, we formulate a QoE-driven resource allocation problem to pinpoint the optimal allocation strategy that maximizes the total QoE over a set of users located behind the same bottleneck link. We do that while accounting for the characteristics of the screens they use for video playout. Our QoE functions were built using curve-fitting on datasets capturing the relationship

between QoE, screen characteristics, and content-related metrics (Bitrate). For this purpose, we use two datasets that link throughput [9] or video bitrate [10] to a QoE level. Using these QoE functions, we propose a simple heuristic based on Lagrangian relaxation and KKT (Karush Kuhn Tucker) conditions to solve the optimization problem efficiently. Our network simulations using ns-3 show that the proposed heuristic can increase overall QoE compared to an allocation with a TCP look-alike strategy implementing max-min fairness.

Our contributions can be summarized as follows:

- We formulate an optimization problem for network resource allocation, which is based on QoE and where QoE functions are built using datasets linking (throughput or bitrate) to MOS.
- We present a relaxation of our problem to a non-linear problem by considering continuous video bitrates. Under this relaxation, we develop a simple and greedy heuristic based on Lagrangian multipliers and KKT conditions and prove that our heuristic converges to a state where all gradients are either equal or constraints on the bitrate reached.
- We use the network simulator ns-3 [120] and an open source implementation of DASH to validate our approach and to propose an implementation of the optimal solution that limits the subset of visible video representations by a player according to the resolution of its viewport.

The rest of this chapter is organized as follows. In Section 5.2 and 5.3 we present our framework and formulate our optimization problem. Section 5.4 shows numerical results and evaluates the gain of the proposed solution compared to other allocation strategies. In Section 5.5 we illustrate our experimental results using the network simulator ns-3, and evaluate the gain in terms of overall QoE for both the optimal solution and practical implementation that limits the video bitrate as a function of the client's viewport. Finally, we conclude our study highlighting the main results.

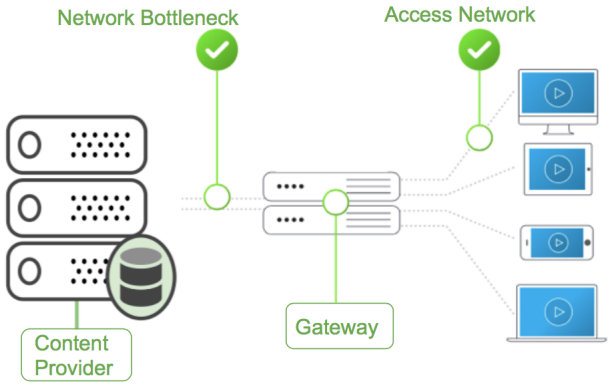


FIGURE 5.1: Framework overview

5.2 Framework and system model

5.2.1 Framework

Consider a set of users with different screen resolutions (alternatively viewports) streaming videos from a server as illustrated in Figure 5.1. Videos on the server are encoded into M different representations (i.e., bitrates or resolutions). We assume users are not limited by their access links and are thus able to download any video representation available on the server. We presume the system's bottleneck to be the backhaul link located between the gateway and the video server. Such a backhaul link can be one of the current wireless networks or the peering link of a network access provider whose users are connected to the Internet by high-speed optical fibers.

In this study, we focus on the problem of QoE-driven bandwidth sharing on the backhaul link and do not consider the presence of any caching functionality at the gateway. Caching would add another interesting dimension to our problem, and would undoubtedly interact with bandwidth sharing on the backhaul, so we differ its joint study with bandwidth allocation to a future work dedicated to the topic. For now, one can see our work as specific to those videos that are not cached. The study of caching is the subject of the following chapter.

5.2.2 System model

We now describe in more detail the model that we consider and introduce our notation. Let F denote the set of video files in our catalog (server) offered to the users. Any video file $f \in F$ is encoded into a set of M representations with f_m being the m-th representation of video f, having an encoding bitrate equal to B_{fm} and corresponding to a particular video resolution. For the sake of simplicity and without loss of generality, we suppose all videos to have the same duration T. Further, we suppose that $\forall f \in F$ and $\forall m \in [1, \ldots, M]$, the video bitrates B_{fm}'s are the same (i.e., which can be seen as the average overall videos of the catalog for representation m). Finally, let S be the vector of distinct screen resolutions.

Notation	Representation
F	Set of videos
M	Number of video representations (resolutions) on the server
S	Set of screen resolutions
λ_f	Request rate per Video f
$\lambda_{f,s}$	Request rate per video f and screen resolution s
α	Parameter of the popularity Zipf distribution
C_l	Backhaul link capacity
$B_{M,s}$	Upper bound on bitrate for $s \in S$
X	Bandwidth allocation vector

TABLE 5.1: Notations of our bandwidth sharing framework

For every $f \in F$ we assign a request rate λ_f (i.e., popularity) according to a Zipf distribution of parameter α. This request rate is the total over the different screen resolutions. Each request to a video f is supposed to originate from a particular screen resolution according to a given probability distribution over S. In practice, a network operator can obtain such information on the originated screen resolution by using the IMEI (International Mobile Equipment Identity) of the end-user device or by collaborating with the video content provider. Multiplied by λ_f, this probability gives the request rate per video f and per screen resolution s that we denote $\lambda_{f,s}$. We have $\lambda_f = \sum_{s \in S} \lambda_{f,s}$. Table 5.1 summarizes the notation used in our framework, while Figure 5.2 illustrates the process of generating requests for the case $|S| = 5$ screens (most common screen resolutions in the mobile market).

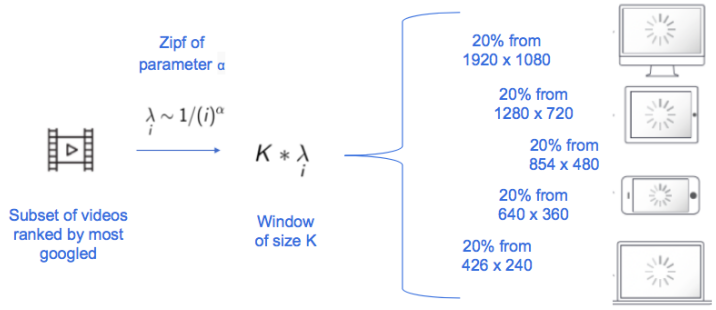


FIGURE 5.2: Traffic generation according to our model

5.2.3 From QoS to QoE

As per prior subjective studies, the QoE of video streaming is a function of application layer QoS features that are either dependent on the video content (e.g., video bitrate) or the playout metrics (e.g., the initial startup delay) [121, 122]; the playout metrics further depend on the underlying network conditions such as the network throughput or delay. In this thesis, we consider building QoE functions that take as input the network throughput or the video bitrate and differ to a future work considering other factors that might also impact the QoE. To build these QoE functions, we rely on two publicly available datasets that map the QoS to the QoE. The first dataset is produced by controlled experiments and links the network throughput to the QoE level according to ITU P.1203 recommendation [9], while the second dataset is based on the work of the Video Quality Experts Group (VQEG) [10] and maps the video bitrate to the MOS. On these datasets, we apply curve fitting methods (e.g., non-linear least squares) with the canonical function given in Equation (5.1) to build our target QoE functions, taking each time as input the network throughput and the video bitrate, respectively. In this Equation (5.1), x stands for the network throughput or the video bitrate, while index s stands for the screen resolution. Constant β is the fitted parameter that determines the shape of the QoE function:

$$QoE_s \underset{exp}{=} QoE_{max}(1 - e^{-\beta_s x}). \tag{5.1}$$

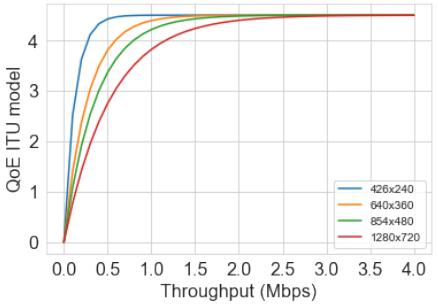


FIGURE 5.3: Fitting QoE function (5.1) using controlled experiments data from [9]

5.2.3.1 From throughput to QoE

The dataset for this model is built by controlled experiments in the lab [9]. The idea behind this dataset is to link the MOS of video streaming to the available bandwidth inside the network. The dataset consists of 100k unique YouTube video playouts under different trace-driven emulated network conditions. This dataset maps the network QoS features such as throughput, delay, and packet loss to application-level measurements such as join time, stalls, and video resolutions. The measured application QoS allows calculating the ITU-T P.1203 subjective MOS [3] for different screen resolutions. We use curve fitting based on Equation (5.1) to establish the relation between the MOS computed according to ITU standard with respect to the sole network throughput for the different screen resolutions as shown in Figure 5.3. The screen resolutions given in the figure correspond to the available video resolutions in the traces of the experiments carried out in [9].

5.2.3.2 From bitrate to QoE

Here we use another dataset to calibrate the QoE function in Equation (5.1), but this time as a function of the video bitrate. Cermak et al. [10] studied the relationship between the video quality, the screen resolution, and the video bitrate using the VQEG datasets. The authors show that for screen resolutions such as CIF, QCIF, and HD, different video bitrates are needed to achieve a certain MOS level (see Figure 5 in [10]). For the main mobile screen resolutions, typically from 426x240 up to 1920x1080, we

[3]The ITU-T P.1203 model is a standardized model that takes as input the application QoS to estimate the subjective MOS.

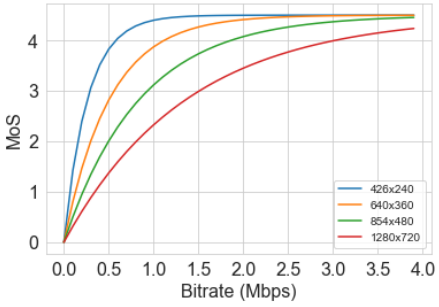


FIGURE 5.4: Fitting QoE function (5.1) using video quality expert group data [10]

extrapolate a vector Z where each entry has two values (z_{BR} , z_{MOS}), then we use curve fitting with Equation (5.1). In Figure 5.4, we plot the fitted curves for four main screen resolutions. These curves, therefore, map the bitrate to the QoE level based on the Video Quality Expert Group (VQEG) data highlighted in [10].

Note that according to [10], the video bitrates take discrete values, thus making the resource allocation problem an INLP (Integer Non-Linear). To counter this difficulty, we relax the problem to make the bitrate take any real value between its minimum and maximum values given by [10], which can transform the resource allocation problem into an NLP (Non-Linear), hence easing the solution. We believe that video content varies considerably in real scenarios, making the bitrate take more diverse values than the ones in [10].

5.2.3.3 Curve fitting evaluation

As highlighted earlier, we rely on two public datasets to map the QoS to the QoE [9, 10]. For each, we use the non-linear least squares method to estimate the parameter β of Equation (5.1) for every screen resolution. We rely on three metrics to estimate the goodness-of-fit of the obtained model: the root mean square error (RMSE), the mean absolute error (MAE), and the R^2 score a.k.a coefficient of determination [123]. In particular, the R^2 score can be seen as the percentage of the prediction value's variance that the model can explain. The closer the R^2 score is to 1, the better the model represents the variance of the dataset. We show in Table 5.2 the estimated parameter β, the mean absolute error, the RMSE, and the R^2 score for every screen resolution using each of the datasets. We note how for curve fitting using the YouTube controlled experiments data,

the RMSE varies between 0.15 and 0.24 on a scale from 1 to 5, indicating high accuracy of the exponential model in Equation (5.1). This good performance of the model is also confirmed for the VQEG data with even lower values for the MAE and the RMSE.

Fitted data	Screen resolution s	Estimated parameter β	MAE	RMSE	R^2
YouTube experiments	426x240	8.17	0.10	0.15	0.91
	640x360	3.73	0.12	0.16	0.92
	850x480	2.75	0.13	0.17	0.92
	1280x720	1.89	0.17	0.24	0.89
VQEG	426x240	3.79	0.13	0.25	0.93
	640x360	1.96	0.02	0.05	0.94
	850x480	1.18	0.07	0.13	0.92
	1280x720	0.72	0.12	0.23	0.93

TABLE 5.2: Curve fitting evaluation

5.3 QoE-driven bandwidth sharing

5.3.1 Problem description

Our problem can be described as follows. Given the different representations of videos, the distribution of screen resolutions, and the backhaul link capacity (denoted C_l), we seek how to share the bandwidth of the backhaul link between the multiple video sessions so that the total system utility, modeled as the overall QoE, is maximized. We want this maximization to account for the screen resolutions and the constraint on the capacity of the backhaul link. Note here that the best we can hope for, from TCP and DASH, is a fair split of the available C_l overall flows except if the client is configured not to download resolutions above some threshold depending on the screen. The optimal allocation is not straightforward, as fairness at the resource level does not necessarily imply fairness at the QoE level. The fact that small screens require lower bitrates than large screens for the same level of QoE is a good illustration.

5.3.2 Problem formulation

Let's introduce the vector $X = (x_s), s \in S$, where the s-th element denotes the bandwidth allocated to each of the users with screen resolution s. The QoE-driven bandwidth

allocation for optimal video quality improvement can be formulated as a nonlinear program (NLP) as follows:

$$\max_{X} \quad U(X) = \sum_{s \in S} \lambda_s QoE_s(x_s) \tag{5.2a}$$

s.t.

$$\sum_{s \in S} \lambda_s T x_s \leq C_l \tag{5.2b}$$

$$x_s \leq B_{M,s}, \qquad s \in S \tag{5.2c}$$

$$x_s \geq 0, \qquad s \in S \tag{5.2d}$$

The global utility function of the system is defined as the sum of weighted QoE functions defined and calibrated in Section 5.2.3.1 and 5.2.3.2. To do so, we aggregate users with the same screen resolution as they are supposed to obtain the same bandwidth allocation ($\lambda_s = \sum_{f \in F} \lambda_{f,s}$). Constraint (5.2b) accounts for the backhaul capacity limitation, whereas constraint (5.2c) upper bounds the allocation for every screen resolution based on the bitrate or throughput needed for excellent video quality at this screen resolution according to the datasets we are using. Note here that this upper bound can be removed as it is accounted for indirectly by the QoE functions (i.e., the QoE is at its maximum value for any allocation greater than this upper bound), but we decided to keep it for clarity of the presentation.

5.3.3 Gradient solution based on Lagrangian relaxation

For the QoE function we consider (Equation (5.1)), our problem is convex and thus possesses a unique solution. One can use well-known heuristics such as Sequential Least Squares [124] to get an approximation of the optimal allocation. The Sequential Least Squares method, part of successive quadratic programming, solves a sequence of optimization sub-problems. For every sub-problem, Sequential Least Squares optimizes a quadratic model of the objective subject to a linearization of the constraints (linear in our case). Given the particular shape of our QoE functions (i.e., mono-variate) and

constraints, and to help to get further insights on the optimal solution, we propose a simple greedy heuristic that helps to approximate the non-linear objective function and efficiently maximizing it. The proposed greedy heuristic considers KKT (Karush Kuhn Tucker) conditions to check if a feasible solution is optimal. We start by writing the partial Lagrangian function obtained by relaxing constraints (5.2c) and (5.2d):

$$L(X,\gamma) = U(X) - \gamma(\sum_{s\in S}(\lambda_s T x_s) - C_l). \tag{5.3}$$

Constant γ is the Lagrangian multiplier associated to constraint (5.2b). By supposing constraint (5.2b) to be set to equality at the optimal solution (otherwise the system is under-utilized), and by differentiating the Lagrangian $L(X,\gamma)$ with respect to allocation vector X, we can prove that a first possible solution could be the one that equalizes all gradients of the QoE functions:

$$\frac{\partial QoE_i(x_i)}{\partial x_i} = \frac{\partial QoE_j(x_j)}{\partial x_j}, \forall (i,j) \in S. \tag{5.4}$$

This, together with $\sum_{s\in S}(\lambda_s T x_s) - C_l) = 0$, gives a system of equations that we denote W and that we can solve to find our first bid on the optimal allocation vector. This first bid is the optimal allocation if the two other constraints (5.2c) and (5.2d) are not violated, otherwise, our vector is not the optimal vector and has to be updated. We use the information on the violated constraints to reshape the search space, i.e., we take those violated constraints one by one, and at each step, we set the corresponding allocation either to zero or to the upper bound, then we replace them in the Lagrangian (5.3) and repeat the previous process until converging to an allocation that satisfies all constraints while nullifying the gradient of the Lagrangian. The following algorithm provides further details on our approach.

Algorithm 1: Compute allocation vector

Output: *Optimal value of* $X = (x_s), s \in S$

Input: $\lambda_s,\ QoE_s,\ B_{M,s},\ C_l$

$$Initialize\ X\ from\ (W) \begin{cases} \frac{\partial QoE_i}{\partial x_i} = \frac{\partial QoE_j}{\partial x_j}\ \forall (i,j) \in S \\ (\sum_{s \in S}(\lambda_s T x_s) - C_l)) = 0 \end{cases}$$

while *either (5.2c) or (5.2d) false* **do**

  if $\exists\ x_i \in X, such\ that\ x_i < 0$ **then**

    Set **Min**$(x_i \in X |\ x_i < 0) = 0\ for\ (X, W)$;

    $Solve\ (W)$;

    Continue;

  end

  if $\exists\ x_i \in X, such\ that\ x_i > B_{M,i}$ **then**

    Set **Max**$(x_i \in X |\ x_i > B_{M,i}) = B_{M,i}\ for\ (X, W)$;

    $Solve\ (W)$;

    Continue;

  end

end

5.3.4 QoE-fairness at the equilibrium

In this section we discuss the QoE-fairness at the equilibrium, as ensured by our optimal solution. We always consider the case when the QoE is modeled using an exponential function, even though the method hereafter applies to other functions. The optimal QoE-aware bandwidth allocation is reached when the gradients of QoE are equal (see Equation (5.4)). Let's consider two screen resolutions (i) and (j), and let's use Equation (5.4) and the QoE definition in Equation (5.1) to derive the relation between the QoE achieved by the two screen resolutions at the equilibrium. After differentiation and substitution of the exponential term by its value as a function of QoE_{max} and (QoE_i, QoE_j), we get the following equality at optimal allocation:

$$\beta_i(QoE_{max} - QoE_i) = \beta_j(QoE_{max} - QoE_j), \qquad \forall i, j, 0 < x_i < B_{M,i}, 0 < x_j < B_{M,j}. \tag{5.5}$$

Note that for this, we suppose that the constraint on the maximum bitrate is not reached. The constraint on the impossibility of a negative bitrate is naturally not reached at optimal allocation. In case the former constraint is reached for some screen resolutions, those screens will be at their maximum QoE, and the equality will only hold for the other screens for which the QoE maximum is still not reached. The above equality implies that for the optimal solution of QoE-aware bandwidth allocation, the difference between the maximum QoE and the achieved QoE is inversely proportional to the β value of the corresponding screen resolution. An example of these β values can be found in Table 5.2. This can be written as:

$$(QoE_{max} - QoE_i) \propto \frac{1}{\beta_i}. \tag{5.6}$$

We can thus conclude that the slower the convergence of the QoE function with the bitrate (i.e., case of large screens), the smaller the β and the smaller the QoE value. On the other hand, the faster the convergence of the QoE function (i.e., case of small screens), the larger the β and the better the QoE value. Small screens thus achieve better QoE than large screens, but the latter ones don't starve either. They still achieve an acceptable QoE level, with the distance to the maximum QoE level inversely proportional to their β value. For example, by referring to Table 5.2, screen resolution 1280x720 is only at twice the distance from maximum QoE than screen resolution 640x360. Note here that it is only the distance to maximum QoE that goes inversely proportional to the β parameter at the optimal allocation of backhaul bandwidth, not the absolute value of the QoE itself.

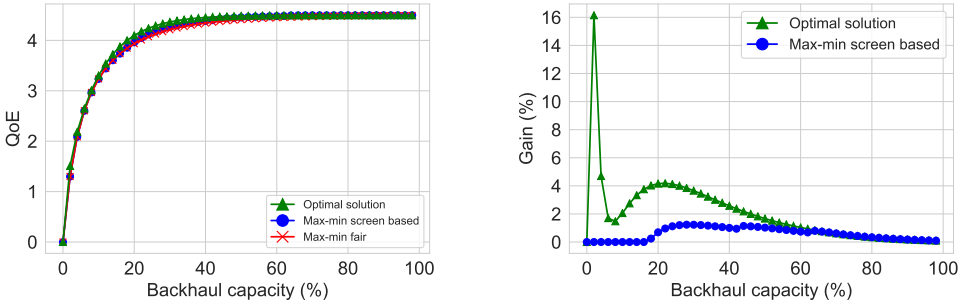
5.4 Numerical simulations

5.4.1 Simulation setup

We consider a network where a set of users have different screen resolutions distributed uniformly over S. We consider S to include the five standard screen resolutions depicted in Figure 5.2. Videos are of equal duration, and the allocation vector X, in this case, is proportional to the number of bytes each video would require from the network.

As reference allocations, we consider two max-min allocations, which model the existing solutions based on TCP and DASH. The first allocation is called *max-min fair* which consists of video flows sharing equally the available bandwidth independently of the characteristics of their screens (i.e., a flow can get more than it can play out). The second allocation is called *max-min screen based* where bandwidth is fairly shared but in the limit of maximum supported bitrate per screen (denoted $B_{M,s}$ according to our notation). This consists of a video flow of screen resolution s fighting for the bandwidth and sharing it fairly with the others as long as the maximum bitrate $B_{M,s}$ is not reached. Once reached, the flow (i.e., DASH) does not ask for higher bitrates even if bandwidth is available in the network. This control can be either implemented at the client or at the server if the information on the screen (or viewport) is made available to it.

In addition to these reference allocations, we use our heuristic to derive the best bandwidth allocation that maximizes the sum of QoE functions over all flows. We show results for the exponential QoE function in Equation (5.1), but we also discuss an extreme case where QoE grows linearly with the throughput or the bitrate. Note that the optimization with a linear QoE function cannot be solved with our heuristic as it corresponds to a linear program. We solve it instead with CPLEX [125] and provide an intuitive interpretation of the results. Note also that we focus on a snapshot problem where we perform the optimization only once before assigning the resources.



(a) QoE per allocation strategy

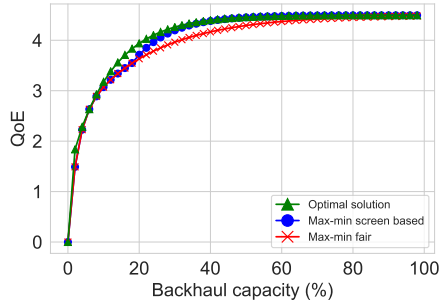
(b) Relative gain, uniform screen probabilities

FIGURE 5.5: Comparison of allocation strategies with uniform screen probabilities (dataset of [10])

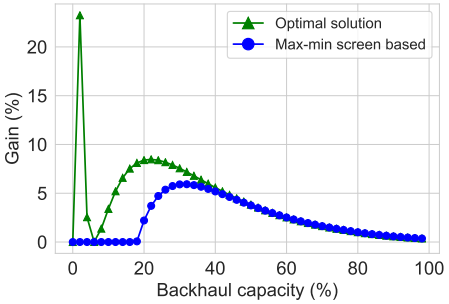
5.4.2 Bandwidth allocation and QoE

We compare the previous allocation strategies in terms of the overall QoE while varying the backhaul capacity. We start by considering the dataset available in [10] (see Section 5.2.3.2) to calibrate our QoE functions. We express the backhaul capacity as a percentage of the worst-case scenario where the operator over-dimensions its network to deliver the maximum bitrate to all users independently of their screen resolutions. Figure 5.5 shows the overall QoE as a function of the backhaul capacity C_l. It also shows the relative gain of the two strategies *optimal* and *max-min screen based* with respect to the baseline strategy *max-min fair*. We notice how leveraging the QoE function with very limited backhaul capacity can achieve a QoE gain up to 16% over the baseline strategy. Furthermore, the *optimal* and the *max-min screen based* strategies manage to reach the maximum possible QoE earlier than *max-min fair*, while increasing the backhaul capacity. For small backhaul capacity (below 20%), *max-min fair* and *max-min screen based* lead to the same result (i.e., zero gain) as the maximum bitrate per screen is not reached, which is not the case of the *optimal* strategy which still delivers a better result. We recall that this result is obtained for screen resolutions of equal popularity over the set S.

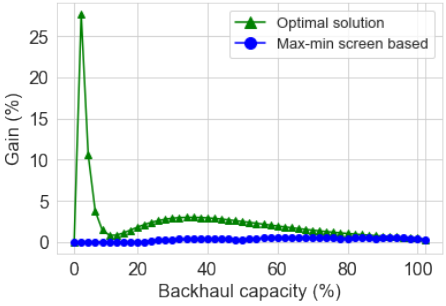
To check for the impact of screen popularity, we apply other distributions of screen resolutions over S and find that our approach works well for other scenarios. Moreover, we point to cases where the gain can actually be up to 20% compared to a simple, fair split of the available backhaul capacity.



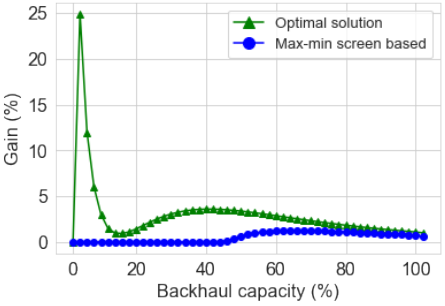
(a) QoE per allocation strategy, 426x240 and 1920x1080



(b) Relative gain, 426x240 and 1920x1080



(c) Relative gain, distribution increasing toward 1920x1080



(d) Relative gain, distribution increasing toward 426x240

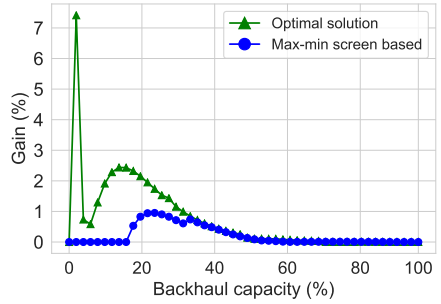
FIGURE 5.6: Comparison of allocation strategies with different screen resolution distributions (dataset of [10])

Indeed, we compare in Figure 5.6 the different allocation strategies for different scenarios. Figures 5.6(a and b) consider the scenario of a bi-modal screen resolution distribution with only two screen resolutions of equal probabilities, 426x240 (small) and 1920x1080 (large). The other two figures, Figures 5.6(c and d), consider a distribution where popularity either increases or decreases with the screen resolution. For example, in Figure 5.6(c), the distribution simulated is 45% of 1920x1080, 35% of 1280x720, 10% of 854x480 and 5% for each of 640x360 and 426x240 screen resolutions. This distribution is then reversed for the simulation highlighted in Figure 5.6(d). We make sure to consider general scenarios and also reflect in some of them the reality of the mobile devices market, where 80% of today's mobile devices have a screen resolution of at least 720p [126] (i.e., the height of the screen or viewport in pixels).

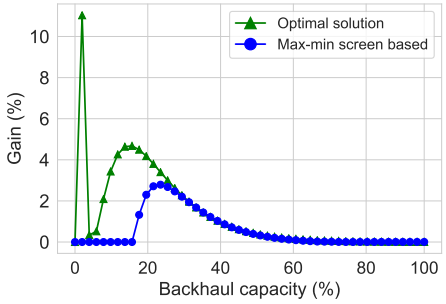
In Figure 5.6(b), we notice how the gain improves and can go above 20%. In this case, we end up with two equal subsets of users, greedy users (large screens) and users easy to satisfy (small screens). In such a scenario, the baseline strategy divides the available

bandwidth fairly among all users, which is insufficient, especially when C_l is small, as large screens cannot get to an acceptable QoE level with the given allocation while small screens get more than needed. However, in Figure 5.6(c and d), we notice that both *max min fair* and *max min screen based* strategies result in almost the same allocation giving approximately the same overall QoE (i.e., the blue line near zero). These two latter figures correspond to scenarios where all screen resolutions are present, but in the first case, the popularity increases with the screen resolution, and in the second case, the popularity decreases with the screen resolution. In the first case, where the larger the screen, the more its popularity, the majority of users are greedy, making it hard for the *max min screen based* strategy to serve all of them even though we are restricting the allocation of small screens. The *screen based* strategy thus behaves approximately as the *max min fair* one. For the second distribution, where the larger the screen, the less popular it is, we end up with many users using small screens and asking for fewer resources. Even though the overall QoE is expected to be better in this case, the fact that the demand is more homogeneous makes the behavior of the *max min fair* strategy again close to the *max min screen based* one. We can notice how for the latter case when the backhaul capacity gets above 40%, small screens reach their upper limit so that the larger screens can get more resources leading to an improved gain. The optimal strategy keeps its good performance over the different scenarios we consider for screen resolution distribution.

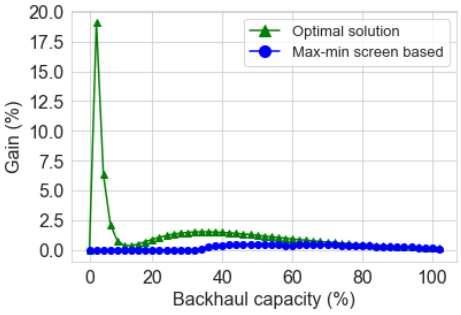
We repeat the same numerical simulation but this time using QoE functions fitted on the controlled experiments data in [9] (see Section 5.2.3.1). Figure 5.7 includes a comparison of the different strategies for the four distributions of screen resolutions: (a) the uniform one, (b) the bi-modal small/large one, (c) the one biased toward large screens, and (d) the one biased toward small screens. We can notice how the gain for these QoE functions drawn according to ITU-T P.1203 standard spans smaller ranges while maintaining the same shape as with QoE functions fitted using the dataset of [10]. In particular, for Figure 5.7(a), where all screen resolutions are uniformly present, our approach results in a gain in overall QoE up to 7% compared to the baseline TCP fair split (*max-min fair*). On the other hand, *max-min screen based* is only 1% better compared to the baseline. Moreover, when 50% of requests come from 426x240 devices and the rest



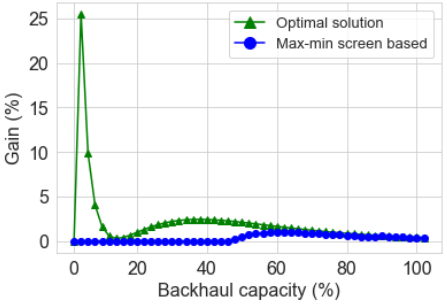
(a) Relative gain, uniform distribution



(b) Relative gain, 426x240 and 1920x1080



(c) Relative gain, distribution increasing toward 1920x1080

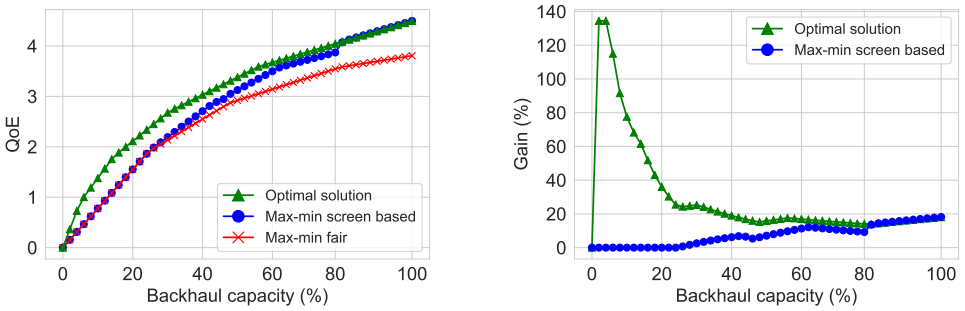


(d) Relative gain, distribution increasing toward 426x240

FIGURE 5.7: Comparison of allocation strategies with different screen resolution distributions (dataset of [9])

from 1920x1080 devices, the optimal allocation (Figure 5.7(b)) results in a gain of 11% compared to baseline allocation. Again we notice that for small C_l, the *max-min screen based* strategy and the *max-min fair* one result in almost the same allocation leading to the same QoE. In Figures 5.7(c and d), we highlight the same behaviour explained earlier, the *max min fair* and the *max min screen based* strategies give close results. The optimal strategy gives better performance, especially for a backhaul capacity of less than 20%.

The above results highlight the interest in the QoE-based approach and show that QoE unaware allocations lead to the best overall QoE neither in restricted backhaul capacity nor in scenarios of homogeneous screen resolutions. Upper limiting the bitrate to the screen resolution works in scenarios of high bandwidth. However, when the bandwidth becomes scarce, it provides close results as the simple screen unaware max-min allocation, thus urging the need for an optimal QoE approach. Finally, we note that even though not considered in this work, our observations will probably apply to larger



(a) QoE per allocation strategy

(b) Relative gain, uniform screen probabilities

FIGURE 5.8: Comparison of allocation strategies with uniform screen probabilities and linear QoE function (dataset of [10])

screens of 2k and more, especially when mixed between each other and with screens of lower resolutions.

5.4.3 Linear QoE

Instead of exponential QoE function, one can imagine an extreme case where QoE grows linearly with the network throughput or the video bitrate, within the range $[0, B_{M,s}]$ for screen resolution s. Even though not realistic, this type of QoE function is interesting because of its implication on the optimal allocation and the way it can be implemented. We add it for completeness of the study, knowing very well that it might not exist in practice. Indeed, in this case, the sum of linear QoE functions transforms the NLP problem into an LP problem that can no longer be solved with our heuristic. Instead, one can use CPLEX to solve it [125]. In such a case, we found that more important gains can be reached. More interestingly, and because slopes of QoE functions are constant but no longer the same for all screens (fast slopes for small screens, slow slops for large screens), the optimal allocation would simply consist of giving full bandwidth priority to small screens on large screens. So small screens are served first in the limit of their $B_{M,S}$, then larger screens, and so on until all screens are served if resources are available. Such allocation, as it requires full priority, cannot be simply implemented on an end-to-end basis (i.e., by limiting the maximum video resolution, for example) but requires the intervention of the network operator as well.

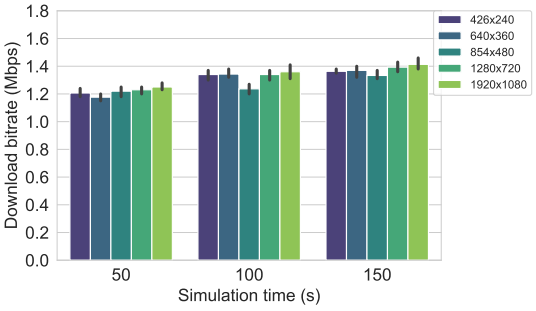
To highlight the above observations, we plot in Figure 5.8 the resulting QoE for the different strategies discussed earlier w.r.t. the percentage of backhaul capacity. Overall, the *max min fair* and *max min screen based* are identical as expected for small backhaul capacity. However, the optimal strategy is giving much more pronounced gains up to 120% compared to the baseline max-min fair strategy.

5.5 Network simulation

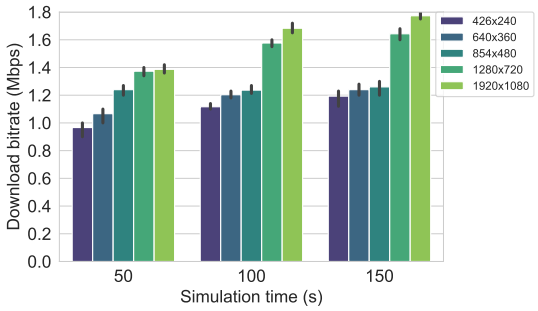
For the validation of our numerical results, we use the simulation software ns-3 [120]. We work with an implementation of MPEG/DASH proposed by [102, 127] that supports the Smooth Video Adaptation Algorithm (SVAA) designed in [101]. DASH being a standard issued by MPEG in 2012 for HAS, different rate adaptation algorithms are proposed in the literature to figure out the resolution of the next segment to download so as to minimize the number of switches and stalls. The SVAA algorithm we consider has shown its efficiency in preventing resolution switches and interruptions [102, 127].

Our simulation setup consists of multiple terminals (15 in total) acting as DASH clients and streaming videos parallel from a DASH server. Clients are connected to a router via access links of 5Mbps and 2ms delay simulating ADSL access links. Their traffic is routed toward the central server through a wired link of fixed capacity 30Mbps and fixed delay 6ms. In terms of video content, we use an animated YouTube video called the Elephants Dream. The video is divided into segments of 2 seconds, produced using the H.264 compression standard with a maximum frame rate of 24 f/s. Moreover, the chunk bitrate is varying from 46Kbps up to 4.3Mbps using traces from [128].

In addition to the standard implementation of DASH/SVAA, we propose two other implementations illustrating the different aspects of our approach. To simulate the optimal solution (see Section 5.3.3), we find the optimal allocation of a video flow using our heuristic then limit the view of the client to video representations not exceeding this allocation. Moreover, we implement screen-based max-min by changing the DASH/SVAA client so that the maximum downloadable representation is the ceil of the maximum bitrate for which the given screen resolution attains the maximum QoE ([10]).



(a) The legacy DASH



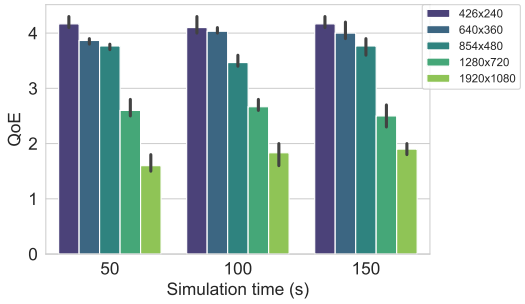
(b) The QoE based DASH

FIGURE 5.9: Average download bitrate per simulation time for each screen resolution over a shared link of capacity $C_l = 30Mbps$

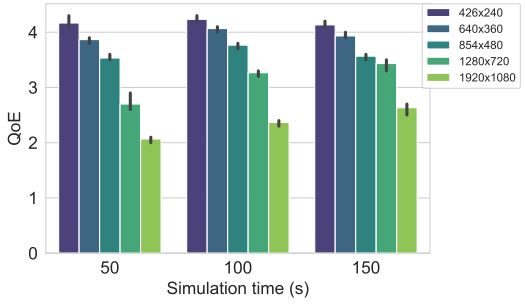
5.5.1 Simulating QoE-driven DASH

We assign to our fifteen devices screen resolutions from a subset of five major mobile screen resolutions (e.g., from 426x240 to 1920x1080) using a uniform probability distribution. The bandwidth allocation itself is calculated at the beginning of the simulation and is maintained constant through the simulation time before being recalculated for the following simulation based on its new configuration. We simulate the users' behavior with each of the described DASH implementations, then we plot the average attainable download bitrate over the shared link per screen resolution and calculate the total corresponding QoE using the functions fitted in Section 5.2.3.2. We repeat every simulation at most 20 times and average outputs to reduce the bias effect and smooth the results. For every average result, the 90% confidence intervals are plotted to help assess their convergence.

Figure 5.9(b) illustrates the effect of QoE-based DASH, where we can see the attainable download bitrate proportional to the screen resolution as needed for good video quality. This is contrary to Figure 5.9(a), where all users grab approximately the same share of



(a) The legacy DASH



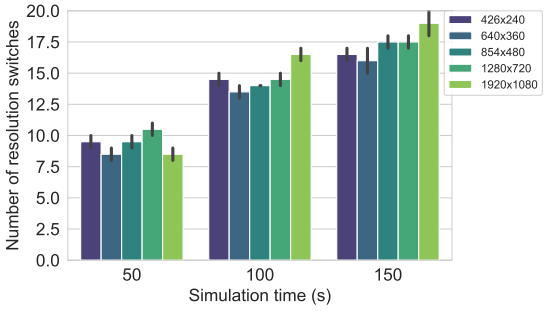
(b) The QoE based DASH

FIGURE 5.10: Average QoE per simulation time for each screen resolution

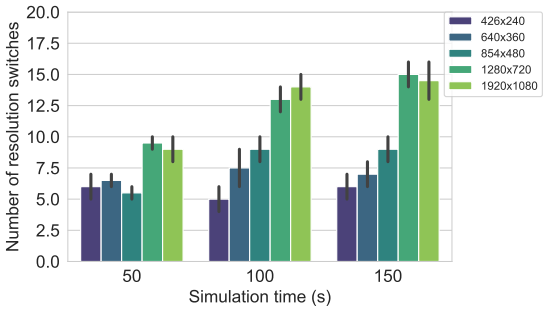
the available capacity as expected with a simple max-min allocation delivered by legacy DASH and TCP.

In Figure 5.10 we compute the average QoE per screen resolution for two implementations: (a) legacy DASH and (b) QoE based DASH. We notice through the figures how the QoE is rearranged in between the different screen resolutions. Thanks to screen resolution consideration, we manage to enhance the average QoE for greedy users (big screen resolution) while maintaining a good QoE level for the others, all this without exceeding the backhaul budget.

We also compare the different implementations in terms of main application-level QoS factors (e.g., stalls, resolution switches) that could impact the subjective QoE [121, 122]. In Figure 5.11, we focus on the quality switches as they appear because of the DASH dynamics. We can see in Figure 5.11(b) how QoE-based DASH manages to reduce the number of switches per screen resolution during a watching session compared to legacy DASH (Figure 5.11(a)). The reduction can be particularly noted for small screens that are now limited at the application level and thus have enough margin at the TCP level to further increase their download rate if needed by fast-moving chunks. This margin does



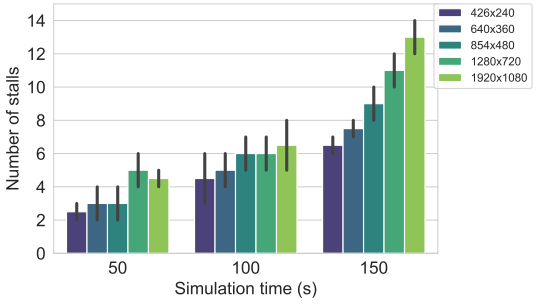
(a) The legacy DASH



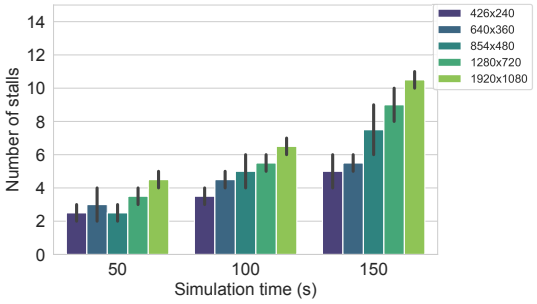
(b) The QoE based DASH

FIGURE 5.11: Average number of resolution switches per simulation time for each screen resolution

not exist for large screens that are throttled by TCP because for their high bitrate, they still achieve fewer resolution switches than in the case of legacy DASH. We also plot the average number of stalls per screen resolution for both the legacy and the QoE-based DASH. In Figure 5.12 we can make the same observation as with resolution switches, the QoE approach reduces the number of video stalls per screen resolution, making the video experience smoother and more satisfying for both small and large screens. At last, we plot in Figure 5.13 the average duration of stalls for the different screen resolutions considered. We include the duration of stalls as prior QoE studies give high interest to user engagement and the ability to tolerate a certain duration of stalls before abandoning the video session. In general, the QoE-based DASH is able to reduce the average duration of stalls, which can be noticed for all screens, in particular, high screen resolutions.

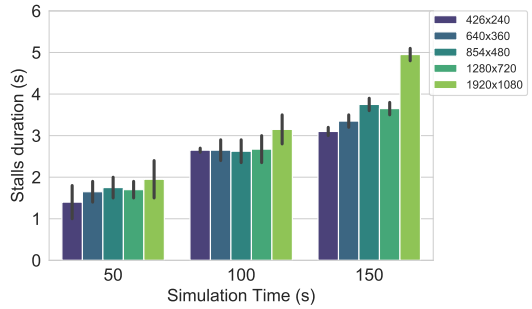


(a) The legacy DASH

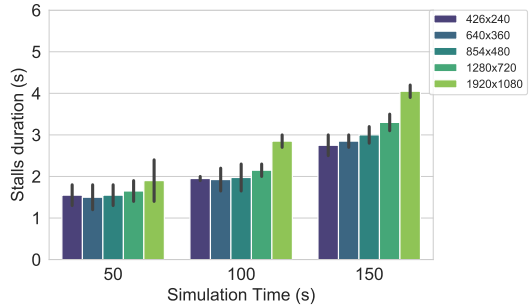


(b) The QoE based DASH

FIGURE 5.12: Average number of video stalls per simulation time for each screen resolution

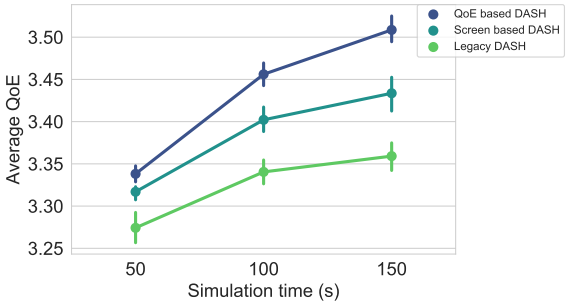


(a) The legacy DASH

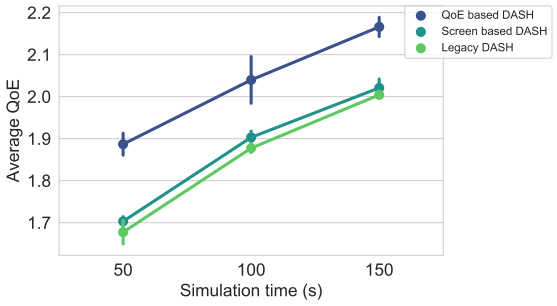


(b) The QoE based DASH

FIGURE 5.13: Average duration of video stalls per simulation time for each screen resolution



(a) Average overall QoE for $C_l = 30$Mbps



(b) Average overall QoE for $C_l = 10$Mbps

FIGURE 5.14: Average overall QoE per simulation time for two backhaul capacities

5.5.2 Changing the backhaul capacity

From the above results, max-min screen-based allocation seems to be an efficient allocation easy to implement and to provides close gain to the optimal allocation. The latter strategy can be implemented, as modifications can be done at the dash.js to limit the maximum downloadable video representation to the maximum screen capacity. However, we expect such allocation to deviate from the optimal when the stress on the backhaul link increases (either more traffic or less bandwidth). Indeed, for more congested scenarios, the fair share of bandwidth of a flow of some screen resolution s becomes likely less than the maximum bitrate for that resolution $B_{M,s}$, which makes the limit on the bitrate driven by the screen resolution less effective. We expect therefore *max-min screen based* to be closer to *max-min fair* and farther from *optimal*. To illustrate this observation, we redo the above ns-3 simulations using a shared capacity of $C_l = 10$Mbps while maintaining the delay to 6ms. Figure 5.14 shows the total QoE for the two cases: (a) $C_l = 30$Mbps and (b) $C_l = 10$Mbps, and this is for the different allocation strategies. In addition, we show 90% confidence intervals for the observed results. In both cases, QoE-based DASH outperforms the other implementations achieving higher overall QoE.

However, for limited backhaul capacity (Figure 5.14(b)) and as expected, screen-based DASH gives approximately the same results as legacy DASH. The results confirm our intuition that allocating based on QoE always helps to shape boundaries of bandwidth space and to improve overall QoE regardless of the available capacity, which is not the case of the other two allocations.

5.6 Conclusion

In this chapter, we studied the problem of bandwidth allocation for multiple video streaming sessions over a shared link. The goal was to maximize the average QoE (Quality of Experience) by leveraging screen resolution. In the first place, we revisited previous studies, able to link either video bitrate [10] or throughput [9] to QoE level for a given screen resolution. We then formulated an optimization problem trying to maximize the overall QoE under linear constraints. To that aim, we proposed a Lagrangian-based solution to approximate the optimal allocation. Later, we showed through numerical and network simulations how leveraging screen characteristics leads to overall QoE improvement in the context of a QoE-driven bandwidth allocation framework. In addition, accounting for screen resolution reduced both switches and interruptions over a watching session.

In the next chapter, we will be studying another aspect of video content management on the Internet, which video caching at the edge. The caching part will validate further the importance of the viewport resolution and the end-users QoE in choosing the video content and video representations to cache.

The contributions related to this chapter appeared in the following publications:

- Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On accounting for screen resolution in adaptive video streaming: QoE-driven bandwidth sharing framework**. – *15th International Conference on Network and Service Management (CNSM)* October 2019, Halifax, Canada

- Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On accounting for screen resolution in adaptive video streaming: QoE-driven bandwidth sharing framework**. – *International Journal of Network Management, Wiley*, May 2020.

Chapter 6

QoE-aware cache placement for adaptive video streaming

In the previous chapter, we proposed QoE models that account for the viewport resolution and use them to reshape the bandwidth allocation and maximize the overall QoE for a set of users streaming videos over the same bottleneck link. In parallel, and to handle the increasing demand for video streaming, service providers resort to deploying edge servers to reduce the rush on their servers, balance the load between them and over the network, and smooth out the traffic variability. The challenge is how to cache video content that maximizes the overall QoE of end-users while accounting for the diversity of their screen resolutions.

Here, we study the viewport aware caching optimization problem for dynamic adaptive video streaming. First, we formulate the proposed optimization problem as an Integer Linear Program (ILP) that balances minimal join time and maximal visual experience, subject to the cache storage capacity. Then, we develop a greedy algorithm to decide on the content to cache using the optimal solution's footprint.

6.1 Introduction

To prioritize or load balance traffic efficiently, caching is a promising solution emerging through the surface, pushing the content to the network edge. In particular, mobile edge caching (MEC) leverages storage capacity within the network to host popular multimedia content, easing video traffic delivery, smoothing its variability, and reducing congestion and access delay [129, 130]. A challenging task is selecting the appropriate video to cache to maximize the overall users' QoE without exceeding the cache storage capacity. In adaptive video streaming, several representations of different and even of the same video will be in competition to be stored, making the selection problem more challenging to solve.

Therefore, we propose a new cache placement optimization framework for adaptive video streaming that accounts for the impact of end-user display capacity and video characteristics (e.g., encoding bitrate and popularity) in addition to the internet access speed. We formulate the optimal cache placement problem as an Integer Linear Program (ILP) aiming to maximize the average QoE over a set of users with the cache storage capacity as a constraint. The optimal solution, using CPLEX [125], can find a selection of videos and representations to cache, ensuring minimal join time and maximal visual experience. Further, we develop a practical greedy caching heuristic using the optimal placement's footprint, offering a near-optimal performance in terms of average QoE per request. Through extensive simulations and different settings, we show that our heuristic outperforms the state of the art caching strategies, which do not account for the device display factors through the placement process. Overall, the main contributions of this chapter can be summarized as follows:

- We formulate the optimal cache placement problem for adaptive streaming in a way to allow caching multiple representations of the same video. The proposed cache placement leverages the users' viewport resolution heterogeneity and allocates the cache size storage based on an objective function reflecting the QoE relation to the video content (bitrate), the application-level QoS (join time), the viewport resolution, and the access speed distribution.

- We propose a near-optimal heuristic called $QoEscoreMax$ to solve the optimization problem in a greedy way. The proposed heuristic uses a metric called $QoEscore$ to rank video representations and decide about caching them or not. This metric incorporates the expected QoE reward resulting from caching a particular representation of a certain video.
- We conduct extensive simulations with multiple settings and show that our heuristic outperforms legacy caching strategies in multiple scenarios in terms of QoE gain, while efficiently exploiting the available cache storage.

The rest of this chapter is organized as follows. In Section 6.2 we present our framework and the notation we used. Then, in Section 6.3, we formulate the optimization problem and highlight the main components of our heuristic. Later, in Section 6.4, we illustrate simulation results for multiple network scenarios and evaluate the gain achieved in terms of overall QoE for different caching strategies. In Section 6.5, we provide a sensitivity analysis of the optimal cache placement w.r.t. join time and encoding bitrate. Finally, we conclude our work.

6.2 Framework and system model

6.2.1 Framework

We consider a single edge cache scenario as depicted in Figure 6.1. The origin server stores a catalog of $\mathcal{F}$ video files, each of which is encoded into $\mathcal{M}$ different representations. We have an edge server able to prefetch video files and cache them in advance. The origin server pushes popular content to the network edge during the off-peak hours, reducing the load on the origin server and resulting in more optimized delay and a more convenient user experience. Usually, content providers put in place several edge servers to be as close as possible to different end-users, and one user can connect to several edge servers at a time. However, in this first study and to confirm the sound of our approach, we consider the case of one edge server. This assumption is similar to considering end-users able to connect to one edge server [98], which is also equivalent to optimize for

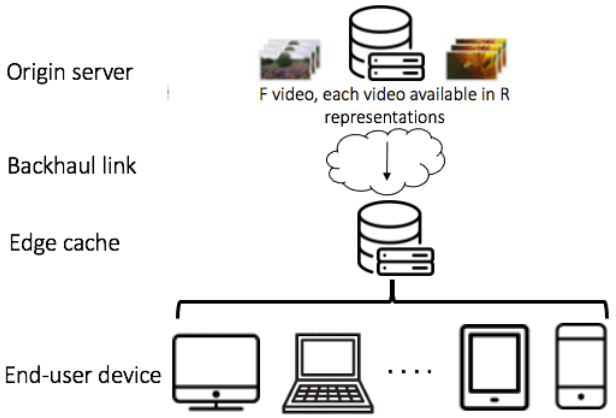


FIGURE 6.1: Framework description

each edge server individually. The case of cross-optimization among edge servers is left for a future study.

In our context, whenever a client wants to play a video, it sends via its DASH client a request to the origin server, which gets redirected to the closest edge server, delivering back the highest video representation available and supported by the client network connection. In case multiple representations of the requested video are available, the edge server will deliver back the one affordable by the client connection and terminal display capacity. Usually, when no representation is found on the edge server, the user request is served directly by the origin server, which will deliver the best video representation affordable by the bottleneck link between the end-user and the origin server.

6.2.2 System model

Here, we discuss in detail the system model and notation used. We consider a catalog of $\mathcal{F}$ video files available on the origin server. Each video $f \in \mathcal{F}$ is available in $\mathcal{M}$ representations, such that $\forall\ m \in \mathcal{M}$ and $\forall\ f \in \mathcal{F}$, $B_{f,m}$ is the encoding bitrate of the representation r of video f. Moreover, we consider the $\mathcal{M}$ representations of each video to be ranked in increasing order of bitrates such that $B_{f,m-1} \leq B_{f,m}, 1 < m \leq |\mathcal{M}|$. We will also assume that all videos have the same duration T. This assumption has often been adopted in the literature for the sake of simplicity and with no loss of

generality [98, 131]. Let E_c be the cache of the edge server, and let S_c be its available cache storage capacity in Bytes. On the other hand, let $\mathcal{D}$ denote the set of users' devices that request videos and that are eligible to communicate with E_c. Each $d \in \mathcal{D}$ reaches E_c with a download rate capacity equal to c_d which we assume to be fully dedicated to the video streaming of the device. Moreover, we denote by v_d the viewport resolution of device d. As for content popularity distribution, we assume it to be stationary over the optimization period, and we consider requests to be independent of each other following the well-known Independent Reference Model. We denote by P_f the popularity of video f and we normalize it in such a way that it becomes equal to the probability that any request issued by any device $d \in \mathcal{D}$ hits video f independently of the other requests [98].

We aim for a cache placement decision to be made by the origin server, or any other controller, in a discrete-time manner. In plain, the problem can be viewed as an on/off process, where during the off periods, the origin server decides about the content to push based on the inferred characteristics from the previous periods (e.g., video popularity and viewport resolution distribution). In terms of end-user viewport resolution (v_d), content providers have access to this information as it is communicated between the DASH client and the DASH server. Moreover, in this thesis, particularly in Chapter 4, we proposed a machine learning and deep packet inspection solution to get such information with different granularity using features calculated on the encrypted video traffic.

6.2.3 QoE modeling

As depicted in Chapter 2, video QoE models in the literature focus mainly on application-level QoS metrics. However, the viewport resolution is also crucial to the perceived visual experience.

6.2.3.1 From bitrate to QoE

First, we capture the relationship between the viewport resolution and the selected video resolution (e.g., encoding bitrate) and the latter's impact on the QoE. As explained in Chapter 5 (Section 5.2.2), we leverage the same exponential QoE function calibrated

offline using an open-source dataset [10]. This model maps the encoding bitrate, z_{BR}, with the perceived user experience, z_{MOS}, for a set of standard viewport resolutions supported by main streaming platforms, typically from 426x240 up to 1920x1080.

$$QoE_{v_d} \underset{exp}{=} QoE_{max}(1 - e^{-\beta_{v_d} x}). \tag{6.1}$$

6.2.3.2 From join time to QoE

We also account for the join time, which is the time it takes the video to start playing out. Such metric has been largely studied, and its impact on the perceived QoE is well documented in the literature (see Chapter 2). We consider a logarithmic model for the impact of the join time on the QoE as proposed in [46]. Equation (6.2) provides a version of this model fitted by the authors using a crowd-sourced dataset of YouTube video streaming. In this equation, $join_d$ is the join time experienced by device d, which can be set to the time needed to fill up the playout buffer on the device. Equation (6.3) provides an estimation of this time using the encoding bitrate of the representation m of video f, $B_{f,m}$, the playout buffer size in seconds, δT, and the user connection speed c_d.

$$QoE_{join_d} = -0.963 * log(join_d + 5.381) + 5, \tag{6.2}$$

$$join_d = \frac{\delta T * B_{f,m}}{c_d}. \tag{6.3}$$

6.3 Viewport aware optimal cache placement

The viewport aware cache placement problem for adaptive video streaming can be described as follows. Given a catalog of videos and the different representations available, the video popularity distribution, the end-user maximum download speed, and most importantly, the end-user viewport resolution, select a set of video representations worthy

of being cached such that the total system utility is maximized while respecting the cache storage capacity constraint. We recall here that the main novelty of our approach is in jointly considering all these factors, in particular, the viewport resolution and the existence of multiple representations per video.

6.3.1 Utility function

For simplicity and without loss of generality, we consider a caching system where a representation of a video file is either fully cached or not cached at all. We assume that any representation can be played out on any viewport, bringing different satisfaction levels at each time. Devices from their side might have different viewport resolutions and different connection speeds. Further, all representations exceeding the resolution of the viewport bring the maximum level of QoE. In this context, which represents better the reality, the decision on the best representations to cache becomes more complex to solve. To reach an optimal solution, we first start by introducing a binary variable $\alpha_{f,m}$ for the action of caching a representation or not. We then complement it with another binary variable per device d called $\gamma^d_{f,m}$ that specifies which representation of video f is served by the cache to device d in case one or more representations of the video are available in the cache. Otherwise, the request is served by the origin server.

$$\alpha_{f,m} = \begin{cases} 1, & \text{if } file(f,m) \textit{ cached} \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

$$\gamma^d_{f,m} = \begin{cases} 1, & \text{if } file(f,m) \textit{ served to } d \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

We define the QoE-driven utility function for a request issued by device d as the average QoE reward overall videos of the catalog while conditioning on the viewport resolution and the device's connection speed d. We write it as a weighted sum of the two QoE

functions defined in Section 6.2.3:

$$Qgain_d = \sum_{f \in \mathcal{F}} P_f \sum_{m \in \mathcal{M}} \gamma_{f,m}^d * (a * QoE_{v_d} + b * QoE_{join_d}). \tag{6.6}$$

(a, b) are system parameters that can be tuned to adjust the importance of each QoE aspect. In plain, one can choose $a \geq b$ if the visual aspect is more important than the start-up delay and vice versa. In our validation later, we make sure to stress test such parameter so as to asses the impact of each QoE aspect and to understand the extent to which it can modify the behavior of the optimal solution.

6.3.2 Problem formulation

The QoE-driven cache placement problem for adaptive streaming can be formulated as an Integer Linear Program (ILP) in the following way:

$$\max_{\alpha,\gamma} \sum_{d \in \mathcal{D}} Qgain_d \tag{6.7}$$

$$\text{subject to: } \sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}} \alpha_{f,m} * B_{f,m} * T \leq S_c, \tag{6.8}$$

$$\sum_{m \in \mathcal{M}} \gamma_{f,m}^d \leq 1, \ \forall f \in \mathcal{F}, \ \forall d \in \mathcal{D}, \tag{6.9}$$

$$\gamma_{f,m}^d \leq \alpha_{f,m}, \ \forall f \in \mathcal{F}, \ \forall m \in \mathcal{M}, \ \forall d \in \mathcal{D}, \tag{6.10}$$

$$\alpha_{f,m} \in \{0, 1\}, \tag{6.11}$$

$$\gamma_{f,m}^d \in \{0, 1\}. \tag{6.12}$$

In this problem formulation, the objective is to maximize the overall QoE reward summed over the set of devices as defined in (6.7) and (6.6), while considering the network conditions, the video characteristics (e.g., popularity and encoding bitrate) and the end-user viewport resolution. The constraint in (6.8) represents the cache size constraint, with $B_{f,m} * T$ being the part of the cache occupied if we cache file (f, m). The constraint in (6.9) makes sure that each device can only download one representation

per cached video. The constraint in (6.10) establishes the relationship between the two decision variables such that a video representation can be served if it is cached. Finally, the constraints in (6.11) and (6.12) define the binary decisions of caching and serving, respectively.

6.3.3 QoEscoreMax

The optimal solution can be derived using a solver (e.g., CPLEX [125]). However, the underlying ILP can be seen as a knapsack problem which is known to be NP-hard. This leads to an exponential computation complexity with a long execution time when the space for the decision variables, particularly the number of videos and representations, becomes significant. To efficiently solve the ILP, we present a greedy heuristic named $QoEscoreMax$ based on the notion of $QoEscore$. The $QoEscore$ is a new metric we introduce to calculate for each video representation the QoE gain that would result from caching it, summed over the set of devices. Following the same reasoning as in Equation (6.6), we write:

$$QoEscore_{f,m} = \sum_{d \in \mathcal{D}} P_f * (a * QoE_{v_d} + b * QoE_{join_d}).$$

$$QoE_d = a * QoE_{v_d} + b * QoE_{join_d}$$

The $QoEscoreMax$ algorithm caches files having the highest $QoEscore$ in the limit of the cache storage. To further account for the cache space occupied by the video file, we normalize the score by the square of its volume in Bytes, $B_{f,m} * T$ [1]. We use the normalized score to rank representations in decreasing order of QoE gain.

By studying the footprint of the optimal solution as solved by CPLEX, we found out that depending on the network conditions and the viewport resolution distribution. We might only need one representation to hit the optimal. However, when the distance between access link speeds increases, the choice is no longer straightforward, and one

[1] The normalization by the square of the volume was shown empirically to provide better results than the normalization by the volume itself.

representation cannot be enough to satisfy everyone. In this scenario, the optimal adds another representation, giving priority to the most popular videos first. Following this optimal footprint, we update $QoEscoreMax$ to limit the number of representations per video. Overall, we iterate over the $QoEscore$ ranked list with three possible options: either replacing, adding, or simply skipping. For instance, for file (f, m), if we don't have the previous representation cached (e.g., $cached_{f,m-1} = 0$), we add the (f, m) representation directly to the cache while increasing the cache occupancy, otherwise, a $delatQoEgain$ is computed between the two cache states: (1) the new representation replaces the previous one, and (2) the new representation is skipped. A positive value of $delatQoEgain$ means that replacing the previous representation is beneficial and so is taken. Otherwise, no action is taken until the following representation of video f is found in the ranked list of $QoEscore$. This heuristic is detailed in Algorithm 2.

Algorithm 2: QoEscoreMax

Result: $Cached$ $-$ $binary\ placement\ list$ $(\mathcal{F}, \mathcal{M})$

$QoEscore(\mathcal{F}, \mathcal{R}), S_c, cache_{occ}, T, B$ $(\mathcal{F}, \mathcal{R})$

while $B_{(f,m)} * T + cache_{occ} \leq S_c$ **do**

  if $cached_{(f,m-1)} = 0$ **then**

    $cached_{(f,m)} = 1$

    $cache_{occ} = cache_{occ} + B_{(f,m)} * T$

  else

    if *deltaQoEgain(file(f,m), file(f,m-1))* ≥ 0 **then**

      $cached_{(f,m)} = 1$

      $cached_{(f,m-1)} = 0$

      $cache_{occ} = cache_{occ} - B_{(f,m-1)} * T$

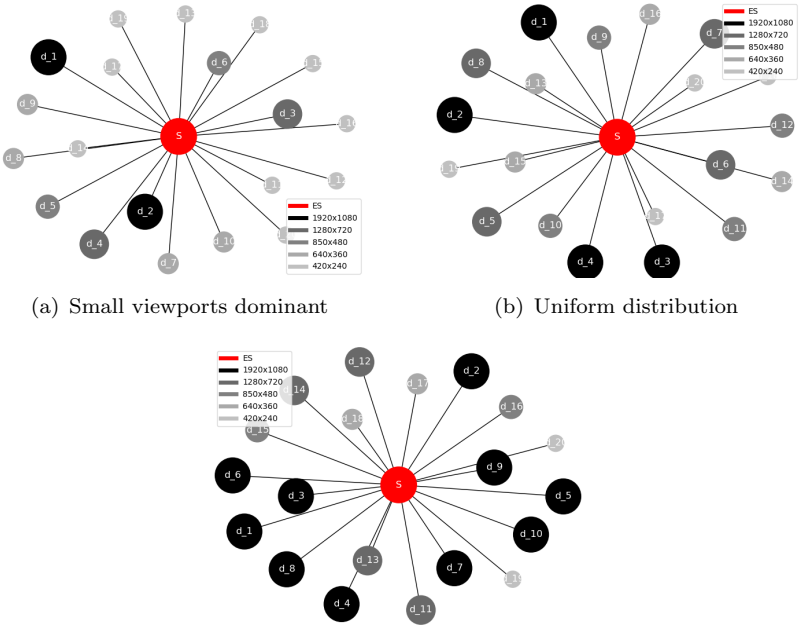
      $cache_{occ} = cache_{occ} + B_{(f,m)} * T$

    end

  end

  $(f, m) = QoEscore.next_{key}$

end



(a) Small viewports dominant

(b) Uniform distribution

(c) Large viewports dominant

FIGURE 6.2: Distribution of devices' viewport resolutions

6.4 Performance evaluation

In this section, we evaluate the performance of our caching framework. To assess the efficiency of our approach, we compare it to a state of the art approaches such as popularity-based caching, which takes into consideration the video popularity [98, 132] and Femtocaching which minimizes the average download delay of video content [93]. Beside our heuristic $QoEscoreMax$, we study different variants of our optimal solution, in particular we show results for *(i)* $ScreenCache$ which does not put any limit on the number of cached representations per video, and *(ii)* $1-rep-ScreenCache$ and $2-rep-ScreenCache$ which limit the number of cached representations per video to a maximum of 1 (i.e. $\sum_{m\in\mathcal{M}} \alpha_{f,m} \leq 1$) and 2 (i.e. $\sum_{m\in\mathcal{M}} \alpha_{f,m} \leq 2$) representations, respectively. For popularity-based caching, we implement a greedy version called $PopularityCache$ that cache representations in increasing bitrate order using the popularity ranking. For $FemtoCache$, we use CPLEX to get its optimal solution leveraging the video popularity and the network conditions.

6.4.1 Simulation settings

We develop a numerical simulator in Python where videos are cached and QoE calculated according to Equation (6.1), (6.2) and (6.3). We consider a network of 20 devices and sample the devices' viewports over a set of standard viewport resolutions (420x240, 640x360, 850x480, 1280x720, 1920x1080), following three main scenarios (Figure 6.2); (1) small viewports dominant, (2) uniform viewports distribution, and (3) large viewports dominant. In terms of network access, we consider two scenarios depicting a case where users have either high download rates (from 10 to 18 Mbps) or poor/medium download rates (from 1 to 7 Mbps). As for video content, we consider a catalog of 20 videos of same duration $T = 60s$, each video is encoded in 7 representations with encoding bitrates (0.25, 0.55, 0.95, 1.5, 2.6, 5, 8 Mbps). We further assume that the popularity of the videos follows a Zipf distribution with parameter 0.56 [131]. Last, the storage capacity is varied as multiple of the average size of a video representation.

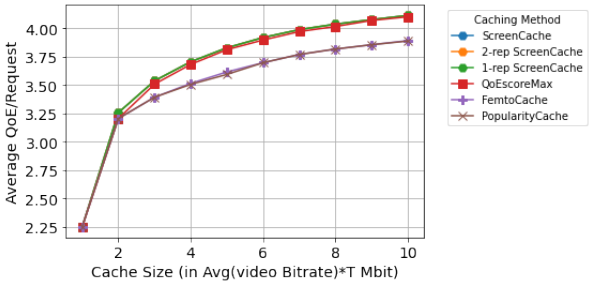
At this stage, we consider $a = b = 0.5$ in Equation (6.6), such that the encoding bitrate and the join time have the same importance on the user experience. To compare the different caching strategies, we use the metric $AverageQoE/request$, representing the average perceived QoE over the set of devices and videos. Each request targeting a random video in the catalog will be potentially served by the cache given the selection of cached representations and following the process described in Section 6.2. This metric, between 0 and 4.5 (maximum QoE), also includes the notion of hit/miss, as the cache misses will result in zero contribution to the QoE_{gain}. A higher $AverageQoE/request$ means that a large proportion of requests result in a cache hit and that the perceived QoE of each hit is relatively good. We don't consider the QoE of downloading from the origin server in case of a miss because we aim at optimizing the cache behavior independently of what provides the internet backbone as download performance.

6.4.2 Simulation results

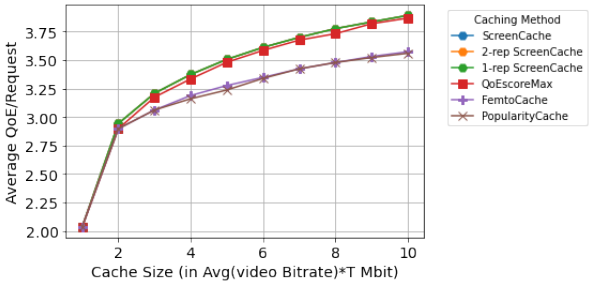
We start with the scenario of high access rates. In Figure 6.3, we plot the $AverageQoE/request$ vs. the cache capacity for the three different viewport distributions. In plain, Figure 6.3 (a), most devices are sampled with small viewports. Here, the optimal $ScreenCache$ derived by the CPLEX results in the same QoE as $1-rep-ScreenCache$ and $2-rep-ScreenCache$, suggesting that the optimal can be achieved with only one representation per video. On the other hand, $FemtoCache$ and $PopularityCache$ perform similarly, and below the optimal, the reason is that $PopularityCache$ by proceeding in increasing order of bitrates ends up giving priority to the smallest representations, which result in almost the same behavior as the $FemtoCache$ scheme which tries to minimize the average file download delay. Meanwhile, $QoEscoreMax$ outperforms the previous two caching strategies and highlights a near-optimal performance. Thanks to using the $QoEscore$ metric, $QoEscoreMax$ prioritizes the most rewarding representations making possible the caching of other than the lowest representation if needed by some viewports and some access links. The lack of viewport resolution notion in $FemtoCache$ and $PopularityCache$ downgrades their performance, especially when the cache size increases as there will be more space to invest in better quality representations to enhance the QoE further.

In Figures 6.3 (b) and 6.3 (c), the same behavior is recorded; however, the average QoE per request is decreasing to 3.75 and 3.5, respectively. The increase in the viewport resolution can explain this. For example, in Figure 6.3 (c), we end up with mostly large viewports. This requires higher representations to scale up the QoE level, which is difficult to achieve at comparable cache sizes. Moreover, moving toward populations watching on large screens, a slight difference between the optimal and $QoEscoreMax$ starts to appear, but most importantly, the gap between optimal and $FemtoCache$ and $PopularityCache$ gets larger. Large screens with good internet access make schemes focusing on low representations or minimizing the file download delay less efficient than the optimal that cache directly those representations providing the maximum QoE gain.

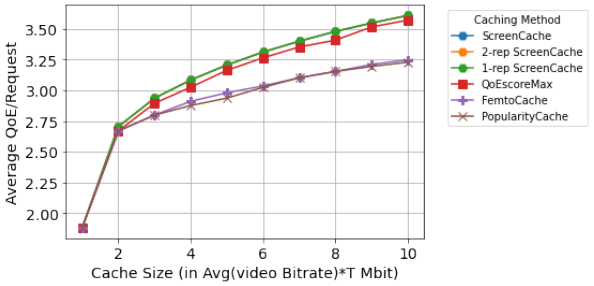
In a second scenario, we consider devices with poor to medium internet access (i.e.,



(a) Small viewports dominant



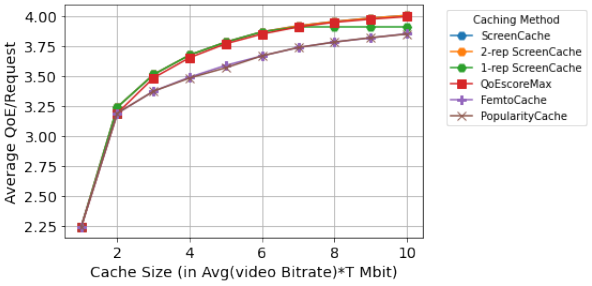
(b) Uniformly distributed viewports



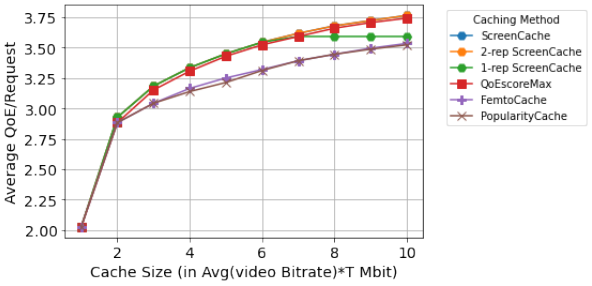
(c) Large viewports dominant

FIGURE 6.3: Average QoE per request for fast internet accesses

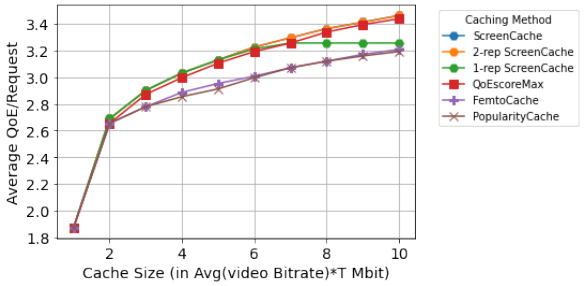
part of devices cannot accommodate all representations). This scenario is more challenging as it requires caching a mix of representations depending on the access speed and the viewport. Here, one can expect $1-rep-ScreenCache$ to diverge from unlimited $ScreenCache$ as the cache size increases. In Figure 6.4, we show results for the three viewport distributions. As we can observe, regardless of the viewport distribution, the $1-rep-ScreenCache$ scheme starts diverging from the optimal as the cache size increases. The task of finding one representation per video that approximates well the optimal for all viewports is no longer possible as some accesses are slow and cannot accommodate high-quality representations. However, we can see that the $2-rep-ScreenCache$ keeps up and shows almost the same behavior as the unlimited optimal. To further understand this behavior, we analyze the footprint of $ScreenCache$



(a) Small viewports dominant



(b) Uniformly distributed viewports



(c) Large viewports dominant

FIGURE 6.4: Average QoE per request for poor/medium accesses

to find out that in our setup, the optimal considers two representations for popular videos while keeping the least popular videos with only one representation of low quality. $QoEscoreMax$ sustains its good performance through the different viewport distributions and outperforms the $1-rep-ScreenCache$ scheme for large cache sizes thanks to its capacity to consider the caching of different representations for popular videos, i.e., Algorithm 2. $FemtoCache$ and $PopularityCache$ fall behind the optimal, with the gap now reduced because of the increased importance of low representations in this scenario.

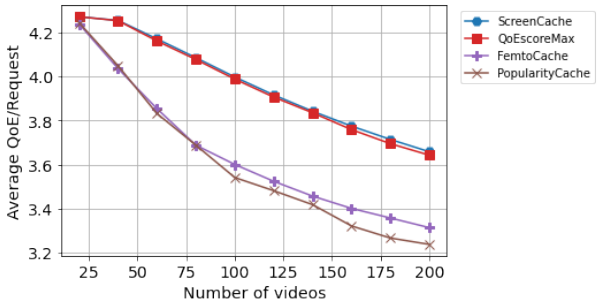


FIGURE 6.5: QoE vs catalog size

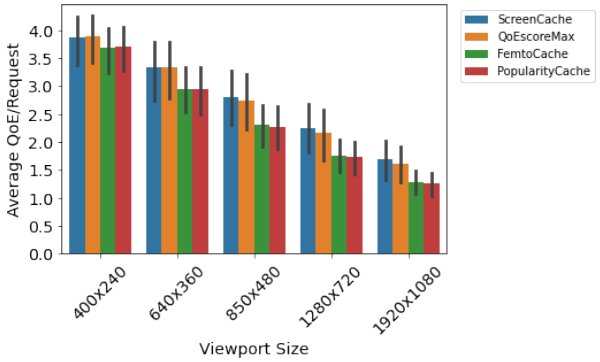


FIGURE 6.6: QoE mainly based on video bitrate

6.4.3 QoEscoreMax vs catalog size

We test the behavior of our solution for a larger video catalog. We plot in Figure 6.5 the average QoE per request for the scenario of good network conditions and large viewports dominant. We scale the catalog size at a fixed cache size (i.e., $10 * Avg(B_{f,m}) * T$ Mbits). Overall, the QoE value is negatively correlated with the catalog size for all caching strategies, making sense since the storage capacity remains the same and the pressure on the cache increases. However, the decline of *ScreenCache* and *QoEscoreMax* is slower than *FemtoCache* and *PopularityCache* as the former can better utilize the available storage by caching the most rewarding content directly in terms of QoE rather than wasting storage on caching low-quality videos from unpopular content.

6.5 Sensitivity analysis

This section evaluates the impact of the parameters used to model the QoE on the cache placement strategy. In particular, the QoE model, in Section 6.3 is composed of two

components with different weights; *(i)* bitrate to QoE (Equation (6.1)) and *(ii)* join time to QoE (Equation (6.2)). Here, we study the impact of this balance between the bitrate and the join time. For space constraints, we only show results for good network conditions and uniform viewport distribution as highlighted in Figure 6.2(b).

6.5.1 Video bitrate over join time

In this part, we give more importance to QoE_{v_d} linking the bitrate to the QoE, with $a = 0.9$ and $b = 0.1$. We plot in Figure 6.6 the average QoE per request (plus its standard deviation) for different viewport resolutions and different caching schemes. These results are calculated over the set of cache sizes shown in Figure 6.3. Overall, we observe that the QoE decreases as we move toward larger viewports. Between the caching schemes, $ScreenCache$ and $QoEscoreMax$ result in almost the same QoE level per viewport resolution, while $FemtoCache$ and $PopularityCache$ fall behind, especially for large screens. This result is in line with what we have observed so far.

6.5.2 Join time over video bitrate

Now, we study the case where the QoE model in Equation (6.6) is mostly based on the join time by considering $a = 0.1$ and $b = 0.9$. Since QoE_{join_d} is negatively correlated with $join_d$ (see Equation (6.2) and (6.3)), one would expect the optimal solution to be selecting representations with smallest encoding rate as they reduce the join time. The model now largely prefers the smoothness of the playout on the quality of the rendered resolution, which is closer in mind to existing placement schemes that seek to minimize the file download delay by caching first the low representations. We illustrate the obtained results in Figure 6.7. Regardless of the viewport resolution, the different caching schemes converge to almost the same QoE level, confirming the selection of the same representations. Overall, this sensitivity analysis validates our approach and shows its good performance and generality when accounting for different aspects of QoE. When the QoE is very sensitive to the join time, it behaves similarly to state of the art schemes by privileging the caching of popular videos of low volume. However, in cases where the QoE is a balanced function of both resolution and join time, state of the art schemes

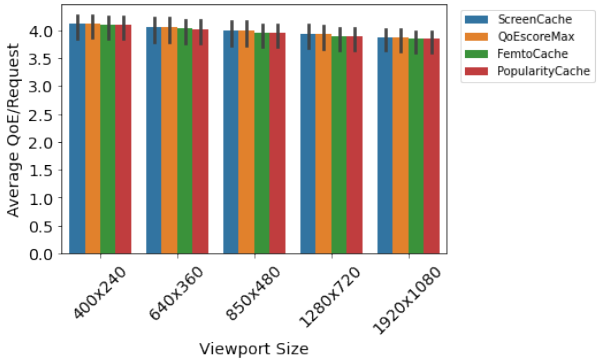


FIGURE 6.7: QoE mainly based on join time

overlook the need to cache high-quality representations for popular content, especially when the viewports are large and the internet accesses are fast enough to accommodate them.

6.6 Conclusion

This chapter studied a QoE-driven cache placement optimization for adaptive video streaming while accounting for the end-user display (viewport resolution). We formulated the problem as an ILP and derived the optimal selection of videos and representations to be cached for different internet accesses and viewport resolution distributions. We also presented $QoescoreMax$, a practical caching heuristic with near-optimal performance. Simulation results showed that our solution strikes the trade-off between optimal QoE and efficient storage management. Moreover, they provided insights on the way to cache the different representations of video content. In good network conditions, one representation can lead to optimal QoE. However, for mild network conditions, the selection process is more challenging as it has to account for the video's popularity before deciding to add a second representation from a video. Meanwhile, based on the importance of the balance between join time and video bitrate, we highlighted rules to select video representations for optimal QoE. Overall, the viewport feedback hints at a more efficient network resource utilization while maximizing the end-user perceived QoE.

The contributions related to this chapter appeared in the following publications:

Othmane Belmoukadam, and Chadi Barakat: **QoE-driven cache placement for adaptive video streaming: minding the viewport**. – *IEEE International Mediterranean Conference on Communications and Networking*, September 2021, Athene, Greece, Hybrid: In-Person and Virtual Conference

Chapter 7

Conclusion and perspectives on future research

7.1 Conclusion

In this thesis, we were mainly interested in understanding video streaming traffic and adequately managing the Internet resources to meet end-user expectations. In a nutshell, we presented an experimental setup and a set of new methodologies to incorporate the end-user viewport resolution in video content management for Internet video streaming. First, we demonstrated in Chapter 3, through controlled experiments, the importance of the viewport resolution and its impact on the observed video resolution patterns and the consumed data budget. Then, in Chapter 4, we proposed a methodology to infer the viewport resolution from the encrypted YouTube video traces using **in-band** network features such as the chunk size and the throughput. Later, we built a QoE model involving the device, media, and network-related factors (e.g., viewport, encoding bitrate, or throughput). Finally, we used the latter models to formulate and solve a bandwidth sharing problem that maximizes the QoE for a set of users streaming over the same bottleneck link (Chapter 5). Moreover, we invested our approach to study a QoE-driven cache placement optimization for adaptive video streaming. In this context, we presented $QoescoreMax$, a practical caching heuristic with near-optimal performance

that highlights the set of video representations worth of caching resulting in maximum QoE reward.

7.2 Limitations

The experimental study highlighted in Chapter 3 on bandwidth waste quantification is related to experiments conducted in good network conditions. In fact, sufficient resource provisioning enables a smooth video playout while supporting higher video resolutions. Our experiments in such a scenario are thus meant to investigate how far the end-user display is considered in video content delivery. However, as shown in Chapter 4, when the network throughput decreases, the impact of the viewport size also decreases, therefore reducing the likelihood of observing a bandwidth waste.

Our methodology for inferring the viewport class and resolution from the YouTube encrypted traces (see Chapter 4) was validated with YouTube encrypted video traces. To that aim, we used tools and open-source documentation provided by YouTube and compatible with the Chrome browser. However, and as our future work suggests, more work can be done to investigate the applicability for other streaming platforms (e.g., Netflix and Dailymotion).

Finally, our QoE modelling highlighted in the Chapter 5, leverages a mono-variate exponential function in line with the IQX hypothesis advanced in [28]. Therefore, it takes as input a QoS parameter related to the media (i.e., bitrate) or the network (i.e., throughput) to determine the QoE. The fact that our exponential function is concave and our constraints are linear (thus convex), we are sure that the considered KKT (Karush Kuhn Tucker) conditions lead to a feasible and unique optimal solution. Furthermore, for our particular case of a single bottleneck link, the analytic study showed that a feasible solution can be obtained by equalizing the gradients of all flows and checking the constraints on the positivity of the allocated bandwidth. This heuristic will have to defined in case of networks of several bottlenecks as the equality of the gradients will not hold overall, but on a network path basis.

7.3 Future works

7.3.1 Studying video streaming in further contexts

We highlighted in Chapter 3 and 4 the importance of the end-user viewport resolution in terms of the observed video resolution patterns and the ability to correlated to **in-band** features. However, when it comes to viewport inference from encrypted video traces, one can extend the work to cover other streaming platforms such as Netflix and Amazon Prime. The latter prepaid services give high importance to privacy and remain very discrete in terms of development kits and assets available for developers seeking to embed their solutions and conduct experiments with their video players. Meanwhile, our approach remains applicable to any streaming platform that provides video player API, data API, and open-source documentation to read the HTTP text messages and derive chunk-related information.

Another direction to extend and validate our work on the viewport is to leverage screens with finer display capacity, such as 4K and 8K. Such a new study can stress test our results with more complex and higher video resolutions. On the other hand, since all our experiments were conducted on desktops and a Chrome browser, an important direction of future work could be to test the same approach and methodology on mobile phones and tablets. According to global statistics provided by the StatCounter website, mobile phones consist of half of the Desktop vs. Mobile vs. Tablet market share worldwide. From this proportion, tech companies like Samsung secured up to 28% followed by Apple with up to 26%. The latter companies equip their mobiles with more and more advanced technologies at each new release. However, working with mobile phones will add another layer of complexity with the end-users mobility and the ability to adapt to cell changes while guarantying a satisfactory user experience. On top of that, mobile phones might have different implementations when it comes to video delivery. The interaction with mobiles and tablets towards experimentation automation and traffic collection is another challenging task.

7.3.2 Dynamic screen-aware bandwidth sharing

In Chapter 5, we proposed and evaluated a framework for optimal allocation of network resources tailored to video streaming applications that allows to maximize the sum of Quality of Experience over all users located behind the same bottleneck link. The particularity of our framework is that it leverages the heterogeneity of screen resolutions to redistribute the network resources, the bandwidth in particular, in a more social way. We carried out an evaluation of our framework both numerically and with ns-3 simulations. Up to now, the optimal allocation was obtained by solving analytically the optimization problem using the heuristic we proposed to this end. For the ns-3 simulations, we proposed an implementation of the optimal solution that upper limits the subset of video representations visible at the players based on the allocation derived from the optimization problem. However, in a real scenario where network traffic is dynamic, and where network traffic not necessarily known beforehand, we would like to reach a level where the network and/or video flows know by themselves their fair share of the available network resources and limit themselves to this fair share. Differently speaking, we will be seeking a distributed implementation of our optimal framework. Such distributed implementation cannot be unfortunately done without the collaboration of the network, as it is the only entity that knows about the actual situation of the traffic and that can send a clear feedback to video clients and video servers to guide them in their search for the equilibrium point. The information about the screen size or the viewport is available at the player, it has to be communicated somehow to the network and the video server on the other side to shape the video streaming traffic. We discuss next our idea about this distributed implementation.

As shown in the problem formulation and the proposed solution, the optimal is the state that equalizes the first degree gradients of all QoE functions. We can thus leverage this property of the optimal solution and imagine a scenario where gradients are transformed into traffic priority, and the network is instructed to treat the video traffic differently based on this priority, in a way to reach equilibrium between the different video flows where their priorities are steered to be equal. By an appropriate binning of the QoE function derivative, we can thus reach levels of QoE that can be used by the network

to penalize clients that are already in low derivative regions meaning they are already having a high QoE. We give priority to clients with high QoE derivative, shifting them up in terms of QoE level. This way, one can see the QoE first order derivative as the marginal gain in terms of QoE for every additional unit of resources allocated by the network to a video flow. This way, the network will help those with high potential gain compared to those with low potential gain, knowing that if a video flow is helped by the network, it will increase its throughput thanks to DASH and TCP, and will then move to a low priority region. Overall, this action should lead to a regime where first derivatives, a.k.a. marginal gains, oscillate around the same value, which is the stable regime we are looking at.

According to our idea, the action taken by the network can be based on priority tags carried by the video packets in their IP packet header. The TOS field of the IP header can be used for this purpose. An efficient way of tagging could be done on the server where the distribution of users and screen resolutions can be tracked in an online fashion without requiring major changes on the client side. Once the server gets the chunk request for some resolution, it labels packets by leveraging the chunk resolution requested, the history of past chunks delivered and the QoE function corresponding to the screen resolution at the client. Labelling could be done by the client as well, but it has to be echoed back by the server, as it has to be carried by data packets in the direction server to client. Now at the network level, packets can be intercepted and treated according to their priority tags by means of mechanisms such as weighted RED [133]. In this case low priority packets will be more affected by the packet drop which latter can be interpreted by the client as a sign of congestion, making it requesting lower chunk resolutions thanks to the closed loop of DASH and TCP. We are currently exploring this venue and studying in particular the interaction between the different control levels that are involved in it. We believe it has sound as a possible implementation of our QoE driven optimization framework.

7.3.3 Collaborative video caching

In Chapter 6, we presented *QoescoreMax*, a caching heuristic with near-optimal performance. The latter solution based its decision on several metrics, including (i) the video bitrate, (ii) the application-level QoS (e.g., stalls and join time), and (iii) the end-user display capacity and (iv) access speed distribution. As highlighted earlier, our solution is optimized for different edge caches separately and does not incorporate any collaboration between caches, whereas today, CDN providers optimize jointly for the entire edge servers.

For future work, one interesting direction is to extend our solution to a cooperative scenario enabling coordination between multiple edge servers orchestrated by a central controller mainly in the context of new network architectures such as Software Defined Networks (SDN) [134]. Today, SDN and network function virtualization (NFV) are at the center of trending networking architectures, as both use network abstractions and softwarization for more flexible network management. The SDN approach centralizes the control and programmability of the network and by default separates the control plane and data forwarding plane. The NFV proposes to virtualize functions like routing and load balancing by transmitting network functions to virtual servers.

We imagine a deployment of our heuristic in a collaborative edge scenario combining both NFV and SDN. An orchestrator, a controller in SDN architectures, performs the video content selection and updates its edge servers' content. To do so, the controller receives periodical updates, and the updates include the information that used in our optimization problem. The end-users connect to the closest edge server and look for the video representation resulting in a maximum QoE reward; if this representation is not there, the edge server will send a request to the controller asking for a redirection to the closest edge server able to answer. The video request will then be redirected to the favorable destination or toward the origin server. The latter edge server keeps a short memory of these misses and where they were forwarded if a similar request arrives and avoids going back to the controller. In SDN, a miss in the match tables triggers a PacketIn message similar to what we are trying to achieve with cache misses [135].

On the edge servers, generally owned by service providers, the storage space is precious and the bill can be heavily affected even though the booked resources might not be fully utilized. In this context, NFV can transform the caching functionality to a virtual function hosted on a server with its virtual cache size. Here, NFV makes the caching functionality more flexible as the virtual cache size can be augmented or reduced based on the load, and the cache instance (i.e., virtual machine) can be migrated from one edge server to another easily. As for forwarding tables, each cache instance will store a table listing the different hosted content with corresponding information like the number of hits per representation. The virtualization of the cache functionality can reduce the bill of content providers and help service providers to benefit from their servers by hosting other network functionalities like load balancing or traffic filtering. Overall, using NFV with the SDN approach eliminates the problems of large and diverse video catalogs. Each edge server does not need to know the closest neighbor nor its content. Instead, it turns to the controller, which aggregates the information forwarded and has a clear vision of the network topology (closest neighbors) and their content.

Chapter 8

Publications

Journal Articles

- Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On accounting for screen resolution in adaptive video streaming: QoE-driven bandwidth sharing framework**. – *International Journal of Network Management, Wiley*, May 2020.

- Othmane Belmoukadam and Chadi Barakat: **Unveiling the end-user viewport resolution from encrypted video traces**. – *IEEE Transactions on Network and Service Management*, May 2021.

Conference Papers

- Othmane Belmoukadam, Thierry Spetebroot and Chadi Barakat: **ACQUA: A user friendly platform for lightweight network monitoring and QoE forecasting**. – *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, February 2019, Paris, France

- Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On accounting for screen resolution in adaptive video streaming: QoE-driven bandwidth sharing framework**. – *15th International Conference on Network and Service Management (CNSM)* October 2019, Halifax, Canada

- Othmane Belmoukadam, Muhammad Jawad Khokhar and Chadi Barakat: **On excess bandwidth usage of video streaming: when video resolution mismatches browser viewport**. – *11th IEEE International Conference on Networks of the Future (NOF)* Octobre 2020, Bordeaux, France.

- Othmane Belmoukadam and Chadi Barakat: **From encrypted video traces to viewport classification**. – *16th International Conference on Network and Service Management (CNSM)*, November 2020, Virtual Conference - BEST PAPER AWARD

- Othmane Belmoukadam, and Chadi Barakat: **QoE-driven cache placement for adaptive video streaming: minding the viewport**. – *IEEE International Mediterranean Conference on Communications and Networking*, September 2021, Athene, Greece, Hybrid: In-Person and Virtual Conference

Posters

- Othmane Belmoukadam and Chadi Barakat: **From encrypted video traces to viewport classification**. – *The World of Industrial Mathematics MOMI2021*, March 2021, Virtual Workshop - BEST POSTER AWARD

Bibliography

[1] Christopher Müller and Christian Timmerer. A vlc media player plugin enabling dynamic adaptive streaming over http. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, page 723–726, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306164. doi: 10.1145/2072298.2072429. URL https://doi.org/10.1145/2072298.2072429.

[2] Thomas Stockhammer. Dynamic adaptive streaming over http –: Standards and design principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, page 133–144, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305181. doi: 10.1145/1943552.1943572. URL https://doi.org/10.1145/1943552.1943572.

[3] Ahmed Mansy, Marwan Fayed, and Mostafa Ammar. Network-layer fairness for adaptive video streams. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2015. doi: 10.1109/IFIPNetworking.2015.7145310.

[4] YouTube. IFrame player API. https://developers.google.com/youtube/iframe_api_reference, 2020.

[5] Google. Chrome Web Request Extension. https://developer.chrome.com/extensions/webRequest, 2020.

[6] Dailymotion. JavaScript SDK. https://github.com/dailymotion/dailymotion-sdk-js, 2020.

[7] Ericsson. Ericsson Mobility Report, June 2018. https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf, 2018.

[8] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67, 2011. doi: 10.1109/MMUL.2011.71.

[9] Muhammad Jawad Khokhar, Thibaut Ehlinger, and Chadi Barakat. From network traffic measurements to qoe for internet video. In *2019 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2019. doi: 10.23919/IFIPNetworking.2019.8816854.

[10] G. Cermak, M. Pinson, and S. Wolf. The relationship among video quality, screen resolution, and bit rate. *IEEE Transactions on Broadcasting*, 57(2):258–262, 2011. doi: 10.1109/TBC.2011.2121650.

[11] IETF. A TCP/IP Tutorial. `https://datatracker.ietf.org/doc/html/rfc1180`, 2021.

[12] HTML. Living Standard. `https://html.spec.whatwg.org/multipage/`, 2021.

[13] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, and et al. The lockdown effect. *Proceedings of the ACM Internet Measurement Conference*, Oct 2020. doi: 10.1145/3419394.3423658. URL `http://dx.doi.org/10.1145/3419394.3423658`.

[14] European Commission. Commission and European regulators calls on streaming services, operators and users to prevent network congestion. `https://ec.europa.eu/digital-single-market/en/news/commission-and-european-regulators-calls-streaming-services-/operators-and-users-prevent-network`, 2020.

[15] Forbes. Netflix Starts To Lift Its Coronavirus Streaming Restrictions. `https://www.forbes.com/sites/johnarcher/2020/05/12/netflix-starts-to-liftits-coronavirus-streaming-restrictions/#7bcba5bf4738`, 2020.

[16] Anush Krishna Moorthy, Lark Kwon Choi, Alan Conrad Bovik, and Gustavo de Veciana. Video quality assessment on mobile devices: Subjective, behavioral

and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6): 652–671, 2012. doi: 10.1109/JSTSP.2012.2212417.

[17] Itu-r recommendation bt.500-11. methodology for the subjective assessment of the quality of television pictures., 2021. `https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-11-200206-S!!PDF-E.pdf`.

[18] G. rubino, "the psqa project," inria rennes-bretagne atlantique, 2021. `http://www.irisa.fr/armor/lesmembres/Rubino/myPages/psqa.html`.

[19] Kandaraj Piamrat, Cesar Viho, Jean-Marie Bonnin, and Adlen Ksentini. Quality of experience measurements for video streaming over wireless networks. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 1184–1189, 2009. doi: 10.1109/ITNG.2009.121.

[20] Kenichiro Masaoka, Takahiro Niida, Miya Murakami, Kenji Suzuki, Masayuki Sugawara, and Yuji Nojiri. Perceptual limit to display resolution of images as per visual acuity. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging XIII*, volume 6806, pages 527 – 535. International Society for Optics and Photonics, SPIE, 2008. URL `https://doi.org/10.1117/12.767143`.

[21] Werner Robitza, Steve Göring, Alexander Raake, David Lindegren, Gunnar Heikkilä, Jörgen Gustafsson, Peter List, Bernhard Feiten, Ulf Wüstenhagen, Marie-Neige Garcia, Kazuhisa Yamagishi, and Simon Broom. Http adaptive streaming qoe estimation with itu-t rec. p. 1203: Open databases and software. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, page 466–471, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351928. doi: 10.1145/3204949.3208124. URL `https://doi.org/10.1145/3204949.3208124`.

[22] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. A bitstream-based, scalable video-quality model for http adaptive streaming: Itu-t p.1203.1. In *2017 Ninth International Conference on*

Quality of Multimedia Experience (QoMEX), pages 1–6, 2017. doi: 10.1109/QoMEX.2017.7965631.

[23] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pages 749–752 vol.2, 2001. doi: 10.1109/ICASSP.2001.941023.

[24] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of youtube qoe via crowdsourcing. In *2011 IEEE International Symposium on Multimedia*, pages 494–499, 2011. doi: 10.1109/ISM.2011.87.

[25] Florian Wamser, Michael Seufert, Pedro Casas, Ralf Irmer, Phuoc Tran-Gia, and Raimund Schatz. Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks. In *2015 European Conference on Networks and Communications (EuCNC)*, pages 239–243, 2015. doi: 10.1109/EuCNC.2015.7194076.

[26] Othmane Belmoukadam, Thierry Spetebroot, and Chadi Barakat. Acqua: A user friendly platform for lightweight network monitoring and qoe forecasting. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 88–93, 2019. doi: 10.1109/ICIN.2019.8685878.

[27] Kamal Deep Singh, Yassine Hadjadj-Aoul, and Gerardo Rubino. Quality of experience estimation for adaptive http/tcp video streaming using h.264/avc. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 127–131, 2012. doi: 10.1109/CCNC.2012.6181070.

[28] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41, 2010. doi: 10.1109/MNET.2010.5430142.

[29] sandvine. The Global Internet Phenomena Report, October 2018. `https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf`, 2018.

[30] Duy Nguyen, J. J. Garcia-Luna-Aceves, and Cedric Westphal. Throughput enabled rate adaptation in wireless networks. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 1173–1178, 2013. doi: 10.1109/ICCNC.2013.6504259.

[31] Ahmed O. El Meligy, Mohamed S. Hassan, and Taha Landolsi. A buffer-based rate adaptation approach for video streaming over http. In *2020 Wireless Telecommunications Symposium (WTS)*, pages 1–5, 2020. doi: 10.1109/WTS48268.2020.9198728.

[32] M. Giordani and M. Mezzavilla and S. Rangan and M. Zorzi. Multi-connectivity in 5g mmwave cellular networks. Mediterranean Ad Hoc Networking Workshop, 2016.

[33] M. Polese and M. Giordani and M. Mezzavilla and S. Rangan and M. Zorzi. Improved handover through dual connectivity in 5g mmwave mobile networks. IEEE Journal on Selected Areas in Communications, 2017.

[34] Video streaming, global market growth, 2021. `https://www.grandviewresearch.com/industry-analysis/video-streaming-market`.

[35] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys Tutorials*, 21(1):562–585, 2019. doi: 10.1109/COMST.2018.2862938.

[36] IETF. Real time streaming protocol (RTSP). `https://tools.ietf.org/html/rfc2326`, 2021.

[37] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages

157–168, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0518-1. doi: 10.1145/1943552.1943574. URL http://doi.acm.org/10.1145/1943552.1943574.

[38] Y. Sani, A. Mauthe, and C. Edwards. Adaptive bitrate selection: A survey. *IEEE Communications Surveys Tutorials*, 19(4):2985–3014, Fourthquarter 2017. ISSN 1553-877X. doi: 10.1109/COMST.2017.2725241.

[39] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, page 97–108, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450317757. doi: 10.1145/2413176.2413189. URL https://doi.org/10.1145/2413176.2413189.

[40] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 272–285, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341936. doi: 10.1145/2934872.2934898. URL https://doi.org/10.1145/2934872.2934898.

[41] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 187–198, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328364. doi: 10.1145/2619239.2626296. URL https://doi.org/10.1145/2619239.2626296.

[42] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016. doi: 10.1109/INFOCOM.2016.7524428.

[43] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE*

Journal on Selected Areas in Communications, 32(4):719–733, Apr 2014. ISSN 0733-8716. doi: 10.1109/jsac.2014.140405. URL `http://dx.doi.org/10.1109/JSAC.2014.140405`.

[44] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 197–210, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346535. doi: 10.1145/3098822.3098843. URL `https://doi.org/10.1145/3098822.3098843`.

[45] ITU-T. Subjective video quality assessment methods for multimedia applications. *ITU-T Recommendation P.910*, 2008.

[46] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*, 2012.

[47] J Gross, J Klaue, H Karl, and A Wolisz. Cross-layer optimization of ofdm transmission systems for mpeg-4 video streaming. *Computer Communications*, 27 (11):1044–1055, 2004. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2004.01.010. URL `https://www.sciencedirect.com/science/article/pii/S0140366404000246`. Applications and Services in Wireless Networks.

[48] Zhou Wang, Ligang Lu, and A. Bovik. Video quality assessment based on structural distortion measurement. *Signal Process. Image Commun.*, 19:121–132, 2004.

[49] Fahad Fazal Elahi Guraya, Ali Shariq Imran, Yubing Tong, and Faouzi Alaya Cheikh. A non-reference perceptual quality metric based on visual attention model for videos. In *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, pages 361–364, 2010. doi: 10.1109/ISSPA.2010.5605523.

[50] Parikshit Juluri, Louis Plissonneau, and Deep Medhi. Pytomo: A tool for analyzing playback quality of youtube videos. In *2011 23rd International Teletraffic Congress (ITC)*, pages 304–305, 2011.

[51] Muhammad Jawad Khokhar, Thierry Spetebroot, and Chadi Barakat. A methodology for performance benchmarking of mobile networks for internet video streaming. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '18, page 217–225, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359603. doi: 10.1145/3242102.3242128. URL https://doi.org/10.1145/3242102.3242128.

[52] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. Measuring the quality of experience of http video streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 485–492, 2011. doi: 10.1109/INM.2011.5990550.

[53] Liu Yitong, Shen Yun, Mao Yinian, Liu Jing, Lin Qi, and Yang Dacheng. A study on quality of experience for adaptive streaming service. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 682–686, June 2013. doi: 10.1109/ICCW.2013.6649320.

[54] Muhammad Jawad Khokhar, Nawfal Abbassi Saber, Thierry Spetebroot, and Chadi Barakat. On active sampling of controlled experiments for qoe modeling. In *Proceedings of the Workshop on QoE-Based Analysis and Management of Data Communication Networks*, Internet QoE '17, page 31–36, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350563. doi: 10.1145/3098603.3098609. URL https://doi.org/10.1145/3098603.3098609.

[55] Linux Traffic Control, 2018. http://lartc.org/.

[56] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. A machine learning approach to classifying youtube qoe based on encrypted network traffic. *Multimedia Tools Appl.*, 76(21):22267–22301, November 2017. ISSN 1380-7501. doi: 10.1007/s11042-017-4728-4. URL https://doi.org/10.1007/s11042-017-4728-4.

[57] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. Measuring the quality of experience of http video streaming. In *12th IFIP/IEEE International*

Symposium on Integrated Network Management (IM 2011) and Workshops, pages 485–492, 2011. doi: 10.1109/INM.2011.5990550.

[58] M. H. Mazhar and Z. Shafiq. Real-time video quality of experience monitoring for https and quic. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018.

[59] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, page 211–224, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450317054. doi: 10.1145/2398776.2398799. URL `https://doi.org/10.1145/2398776.2398799`.

[60] Yining Qi and Mingyuan Dai. The effect of frame freezing and frame skipping on video quality. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 423–426, 2006. doi: 10.1109/IIH-MSP.2006.265032.

[61] Tahir Nawaz Minhas and Markus Fiedler. Impact of disturbance locations on video quality of experience. EuroITV 2011 Workshop: Quality of Experience for Multimedia Content Sharing, 2011.

[62] B. Lewcio, B. Belmudez, A. Mehmood, M. Wältermann, and S. Möller. Video quality in next generation mobile networks — perception of time-varying transmission. In *2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6, May 2011. doi: 10.1109/CQR.2011.5996089.

[63] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 463–472, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0616-4. doi: 10.1145/2072298.2072359. URL `http://doi.acm.org/10.1145/2072298.2072359`.

[64] Meteor: Free internet speed & app performance test, 2018. `https://meteor.opensignal.com/`.

[65] Rtr-nettest, 2018. https://www.netztest.at/en/.

[66] Ookla. speedtest.net., 2020. http://www.speedtest.net/.

[67] Junxian Huang, Cheng Chen, Yutong Pei, Zhaoguang Wang, Zhiyun Qian, Feng Qian, Birjodh Tiwana, Qiang Xu, Z Mao, Ming Zhang, et al. Mobiperf: Mobile network measurement system. http://www.mobiperf.com/, 2011.

[68] Sensorly - unbiased, real-world mobile coverage, 2020. https://www.sensorly.com/.

[69] Thuy T. T. Nguyen, Grenville Armitage, Philip Branch, and Sebastian Zander. Timely and continuous machine-learning-based classification for interactive ip traffic. *IEEE/ACM Transactions on Networking*, 20(6):1880–1894, 2012. doi: 10.1109/TNET.2012.2187305.

[70] Ratanang Thupae, Bassey Isong, Naison Gasela, and Adnan Abu-Mahfouz. Machine learning techniques for traffic identification and classification in sdwsn: A survey. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018.

[71] Jun Zhang, Chao Chen, Yang Xiang, Wanlei Zhou, and Yong Xiang. Internet traffic classification by aggregating correlated naive bayes predictions. In *IEEE Transactions on Information Forensics and Security*, 2013.

[72] Opendpi. . http://www.opendpi.org/, 2012.

[73] Jawad Khalife, A. Hajjar, and Jesús Esteban Díaz Verdejo. Performance of opendpi in identifying sampled network traffic. *J. Networks*, 8:71–81, 2013.

[74] LiJuan Zhang, DongMing Li, Jing Shi, and JunNan Wang. P2p-based weighted behavioral characteristics of deep packet inspection algorithm. In *International Conference on Computer, Mechatronics, Control and Electronic Engineering*, 2010.

[75] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi, and S. Kianian. Real-time traffic classification based on statistical and payload content features. In *2nd International Workshop on Intelligent Systems and Applications*, 2010.

[76] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape. Portload: Taking the best of two worlds in traffic classification. In *INFOCOM IEEE Conference on Computer Communications Workshops*, 2010.

[77] Chunzhi Wang, Xin Zhou, Fangping You, and Hongwei Chen. Design of p2p traffic identification based on dpi and dfi. In *2009 International Symposium on Computer Network and Multimedia Technology*, pages 1–4, 2009. doi: 10.1109/CNMT.2009.5374577.

[78] Gianluca La Mantia, D. Rossi, A. Finamore, M. Mellia, and M. Meo. Stochastic packet inspection for tcp traffic. *2010 IEEE International Conference on Communications*, pages 1–6, 2010.

[79] A. Rao and P. Udupa. A hardware accelerated system for deep packet inspection. In *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2010.

[80] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. Measuring video qoe from encrypted traffic. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 513–526, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345262. doi: 10.1145/2987443.2987459. URL `https://doi.org/10.1145/2987443.2987459`.

[81] Andrew Reed and Michael Kranch. Identifying https-protected netflix videos in real-time. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, page 361–368, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450345231. doi: 10.1145/3029806.3029821. URL `https://doi.org/10.1145/3029806.3029821`.

[82] Feng Li, Jae Won Chung, and Mark Claypool. Silhouette: Identifying youtube video flows from encrypted traffic. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '18, page 19–24, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357722. doi: 10.1145/3210445.3210448. URL `https://doi.org/10.1145/3210445.3210448`.

[83] Ricardo Matos, Nuno Coutinho, Carlos Marques, Susana Sargento, Jacob Chakareski, and Andreas Kassler. Quality of experience-based routing in multi-service wireless mesh networks. In *2012 IEEE International Conference on Communications (ICC)*, pages 7060–7065, 2012. doi: 10.1109/ICC.2012.6364944.

[84] Sebastiaan Laga, Thomas Van Cleemput, Filip Van Raemdonck, Felix Vanhoutte, Niels Bouten, Maxim Claeys, and Filip De Turck. Optimizing scalable video delivery through openflow layer-based routing. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–4, 2014. doi: 10.1109/NOMS.2014.6838378.

[85] Pham Tran Anh Quang, Kandaraj Piamrat, Kamal Deep Singh, and César Viho. Video streaming over ad hoc networks: A qoe-based optimal routing solution. *IEEE Transactions on Vehicular Technology*, 66(2):1533–1546, 2017. doi: 10.1109/TVT.2016.2552041.

[86] Giacomo Calvigioni, Ramon Aparicio-Pardo, Lucile Sassatelli, Jeremie Leguay, Paolo Medagliani, and Stefano Paris. Quality of experience-based routing of video traffic for overlay and isp networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 935–943, 2018. doi: 10.1109/INFOCOM.2018.8485954.

[87] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321839. doi: 10.1145/2491172.2491181. URL `https://doi.org/10.1145/2491172.2491181`.

[88] Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Tom Bostoen, and Filip DeundefinedTurck. Software-defined network-based prioritization to avoid video freezes in http adaptive streaming. *Netw.*, 26(4):248–268, July 2016. ISSN 0028-3045. doi: 10.1002/nem.1931. URL `https://doi.org/10.1002/nem.1931`.

[89] A. Bentaleb and A.C. Begen and R. Zimmermann. Sdndash: Improving qoe of http adaptive streaming using software defined networking. ACM Multimedia, 2016.

[90] Jan Willem Kleinrouweler, Britta Meixner, and Pablo Cesar. Improving video quality in crowded networks using a dane. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, page 73–78, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350037. doi: 10.1145/3083165.3083167. URL `https://doi.org/10.1145/3083165.3083167`.

[91] Stefano D'Aronco, Laura Toni, and Pascal Frossard. Price-based controller for quality-fair http adaptive streaming. In *2016 IEEE International Symposium on Multimedia (ISM)*, pages 113–118, 2016. doi: 10.1109/ISM.2016.0030.

[92] Dong Liu, Binqiang Chen, Chenyang Yang, and Andreas F. Molisch. Caching at the wireless edge: design aspects, challenges, and future directions. *IEEE Communications Magazine*, 54(9):22–28, Sep 2016. ISSN 0163-6804. doi: 10.1109/mcom.2016.7565183. URL `http://dx.doi.org/10.1109/MCOM.2016.7565183`.

[93] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire. Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory*, 59(12):8402–8413, 2013. doi: 10.1109/TIT.2013.2281606.

[94] Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. Approximation algorithms for mobile data caching in small cell networks. *IEEE Transactions on Communications*, 62(10):3665–3677, 2014. doi: 10.1109/TCOMM.2014.2351796.

[95] A. Sengupta, R. Tandon, and O. Simeone. Cache aided wireless networks: Tradeoffs between storage and latency. *2016 Annual Conference on Information Science and Systems (CISS)*, pages 320–325, 2016.

[96] W. Zhang, Y. Wen, Z. Chen, and A. Khisti. Qoe-driven cache management for http adaptive bit rate streaming over wireless networks. *IEEE Transactions on Multimedia*, 15(6):1431–1445, 2013. doi: 10.1109/TMM.2013.2247583.

[97] Ke Liang, Jia Hao, Roger Zimmermann, and David K. Y. Yau. Integrated prefetching and caching for adaptive video streaming over http: An online approach. In *Proceedings of the 6th ACM Multimedia Systems Conference*, 2015.

[98] Yichao Jin, Yonggang Wen, and Cedric Westphal. Optimal transcoding and caching for adaptive streaming in media cloud: an analytical approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):1914–1925, 2015. doi: 10.1109/TCSVT.2015.2402892.

[99] Guanyu Gao, Weiwen Zhang, Yonggang Wen, Zhi Wang, and Wenwu Zhu. Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns. *IEEE Transactions on Multimedia*, 17(8):1286–1296, 2015. doi: 10.1109/TMM.2015.2438713.

[100] Danny H. Lee, Constantine Dovrolis, and Ali C. Begen. Caching in http adaptive streaming: Friend or foe? In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, page 31–36, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327060. doi: 10.1145/2597176.2578270. URL `https://doi.org/10.1145/2597176.2578270`.

[101] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, page 109–120, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450317757. doi: 10.1145/2413176.2413190. URL `https://doi.org/10.1145/2413176.2413190`.

[102] Dimitrios J. Vergados, Angelos Michalas, Aggeliki Sgora, Dimitrios D. Vergados, and Periklis Chatzimisios. Fdash: A fuzzy-based mpeg/dash adaptation algorithm. *IEEE Systems Journal*, 10(2):859–868, 2016. doi: 10.1109/JSYST.2015.2478879.

[103] Chenghao Liu, Imed Bouazizi, M. Hannuksela, and M. Gabbouj. Rate adaptation for dynamic adaptive streaming over http in content distribution network. *Signal Process. Image Commun.*, 27:288–311, 2012.

[104] YouTube. Video Catalogue. `https://drive.google.com/open?id=1tu0sBInt8xJ9Zn32IDlh6DW_ju2FhEou`, 2020.

[105] Dailymotion. video catalogue. `https://github.com/asap-code/Dailymotion-video-catalogue.git`, 2020.

[106] Social blade. Dailymotion top viewed channels. `https://socialblade.com/dailymotion/top/category/news`, 2020.

[107] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje. The latest open-source video codec vp9 - an overview and preliminary results. Picture Coding Symposium (PCS), 2013.

[108] Walter Fischer. Video coding (mpeg-2, mpeg-4/avc, hevc). Signals and Communication Technology book series (SCT), 2020.

[109] Dailymotion. Dailymotion video encoding recommendation. `https://faq.dailymotion.com/hc/en-us/articles/203655666-Encoding-parameters`, 2020.

[110] Omer Nawaz, Tahir Nawaz Minhas, and Markus Fiedler. Qoe based comparison of h.264/avc and webm/vp8 in an error-prone wireless network. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1005–1010, 2017. doi: 10.23919/INM.2017.7987426.

[111] R. R. Pastrana-Vidal, J. C. Gicquel, C. Colomes, and H. Cherifi. Sporadic frame dropping impact on quality perception. Proc. SPIE 5292, Human Vision and Electronic Imaging IX, 2004.

[112] Statcounter. Desktop screen resolution stats worldwide, 2021. `https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide`.

[113] YouTube. Itag documentation. `https://www.genyt.xyz/formats-resolution-youtube-videos.html`, 2020.

[114] YouTube. Itag Git catalogue. `https://gist.github.com/sidneys/7095afe4da4ae58694d128b1034e01e2`, 2020.

[115] Dailymotion. JavaScript SDK. `https://github.com/dailymotion/dailymotion-sdk-js`, 2020.

[116] ITU-T Rec. P.1203 Standalone Implementation. https://github.com/itu-p1203/itu-p1203, 2020.

[117] Ricardo R. Pastrana-Vidal, Jean Charles Gicquel, Catherine Colomes, and Hocine Cherifi. Sporadic frame dropping impact on quality perception. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging IX*, volume 5292, pages 182 – 193. International Society for Optics and Photonics, SPIE, 2004. doi: 10.1117/12.525746. URL https://doi.org/10.1117/12.525746.

[118] Python. Random Forest classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2020.

[119] Meduim blog. Hyperparameter Tuning the Random Forest in Python. https://towardsdatascience.com/, 2020.

[120] ns3. network simulation. http://www.nsnam.org, 2020.

[121] Muhammad Jawad Khokhar, Nawfal Abbassi Saber, Thierry Spetebroot, and Chadi Barakat. An Intelligent Sampling Framework for Controlled Experimentation and QoE Modeling. *Computer Networks*, 147:246–261, December 2018. doi: 10.1016/j.comnet.2018.10.011. URL https://hal.inria.fr/hal-01906145.

[122] Avşar Asan, Werner Robitza, Is-haka Mkwawa, Lingfen Sun, Emmanuel Ifeachor, and Alexander Raake. Impact of video resolution changes on qoe for adaptive video streaming. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 499–504, 2017. doi: 10.1109/ICME.2017.8019297.

[123] Python. Curve fit evaluation. https://scikit-learn.org/stable/modules/modelevaluation.html, accessed 2021.

[124] Python. Sequential least squares. https://docs.scipy.org/doc/scipy-0.18.1/reference/optimize.minimize-slsqp.html#optimize-minimize-slsqp, accessed 2021.

[125] IBM. Cplex. https://www.ibm.com/products/ilog-cplex-optimization-studio, accessed 2021.

[126] DeviceAtlas. The mobile web intelligence. `https://discover.deviceatlas.com/1145767/thank-you/`, accessed 2021.

[127] Dimitrios J. Vergados, Angelos Michalas, Aggeliki Sgora, and Dimitrios D. Vergados. A control-based algorithm for rate adaption in mpeg-dash. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, pages 438–442, 2014. doi: 10.1109/IISA.2014.6878834.

[128] Video traces. `http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/ElephantsDream/`, accessed 2021.

[129] M. A. Maddah-Ali and U. Niesen. Fundamental limits of caching. *IEEE Transactions on Information Theory*, 60(5):2856–2867, 2014. doi: 10.1109/TIT.2014.2306938.

[130] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018. doi: 10.1109/JIOT.2017.2750180.

[131] Chenglin Li, Laura Toni, Junni Zou, Hongkai Xiong, and Pascal Frossard. Qoe-driven mobile edge caching placement for adaptive video streaming. *IEEE Transactions on Multimedia*, PP:1–1, 09 2017. doi: 10.1109/TMM.2017.2757761.

[132] Wenjie Li, Sharief M. A. Oteafy, and Hossam S. Hassanein. Streamcache: Popularity-based caching for adaptive streaming over information-centric networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016. doi: 10.1109/ICC.2016.7511583.

[133] IETF. Weighted random early detection(wred). `https://tools.ietf.org/html/rfc2309`, accessed 2021.

[134] Sanjeev Singh and Rakesh Kumar Jha. A survey on software defined networking: Architecture for next generation network. *Journal of Network and Systems Management*, 25(2):321–374, Sep 2016. ISSN 1573-7705. doi: 10.1007/s10922-016-9393-9. URL `http://dx.doi.org/10.1007/s10922-016-9393-9`.

[135] Flowgrammable, improving adoption of software-defined networks and networking, 2021. http://flowgrammable.org/sdn/openflow/message-layer/packetin/.