UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ROBERTO IRAJA TAVARES DA COSTA FILHO

From 2D to Next Generation VR/AR Videos: Enabling Efficient Streaming via QoE-aware Mobile Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Luciano Paschoal Gaspary

Porto Alegre
April 2019

CIP — CATALOGING-IN-PUBLICATION

Tavares da Costa Filho, Roberto Iraja

From 2D to Next Generation VR/AR Videos: Enabling Efficient Streaming via QoE-aware Mobile Networks / Roberto Iraja Tavares da Costa Filho. – Porto Alegre: PPGC da UFRGS, 2019.

176 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Luciano Paschoal Gaspary.

1. Quality of Service. 2. Quality of Experience. 3. Video Streaming. 4. Virtual Reality. 5. VR Video. 6. Path Selection. 7. Mobile Networks. I. Gaspary, Luciano Paschoal. II. Título.

"Science is a way of thinking
much more than it is a body of knowledge."
— Carl Sagan

ACKNOWLEDGMENTS

I would like to thank my family, especially my parents, my in-laws, my wife Francine and my son Vicente, for their unconditional love and for bringing happiness into my life.

I would like to express my sincere gratitude to my advisor Prof. Luciano Paschoal Gaspary, for the support, availability and for sharing his knowledge along this path. He taught me more than I could ever give him credit for here.

I would like to thank my colleagues at UGENT, in particular, my advisor abroad Prof. Filip De Turck for hosting me incredibly well during my time in Belgium.

I deeply thank my colleagues at Netmetric, in especial Prof. Valter Roesler, for the opportunity to contribute in such an amazing project.

My sincere thanks also go to all the professors and students of the Computer Networks Group at UFRGS. My research would not have been the same without their support.

Finally, none of this research work would have been possible without the financial support of the Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense (IFSul) and Coordenação de Aperfeicoamento de Pessoal de Nível Superior (CAPES). Thanks for the support and for the opportunity to conduct this research.

ABSTRACT

Ranging from traditional video streaming to Virtual Reality (VR) videos, the demand for video applications to mobile devices is booming. To deal with the massive traffic produced by video applications, mobile operators rely on offloading technologies such as Small Cells, Content Delivery Networks and, shortly, Cloud Edge and 5G Device to Device communications. Although these techniques are fundamental for improving network efficiency, they produce a multitude of paths through which the user traffic can be forwarded. Most importantly, such an increased path diversity does not provide any guarantee regarding user's Quality of Experience (QoE). Thus, a critical problem arises about how to handle the increasing video traffic while managing the interplay between infrastructure optimization and QoE. Solving this issue is remarkably difficult, and recent investigations do not consider the large-scale context of mobile operator networks. In a nutshell, the problem of dynamic provisioning of QoE-aware paths can be decomposed into two fundamental functions: (*i*) QoE measurement or estimation and (*ii*) path selection on a programmable network. To address the problem of QoE estimation, we propose a model to predict video streaming quality based on the observation of performance indicators of the underlying IP network. To accomplish this objective, the proposed model leverages lightweight active measurements and machine learning techniques. In a further step, we introduce a novel QoE-aware path deployment heuristic for large-scale SDN-based mobile networks. The scheme relies on both a polynomial-time algorithm for composing multiple QoS metrics and a scalable QoS to QoE translation strategy. Obtained results show that the proposed methods for video streaming performance prediction produce accurate estimates. As a consequence, our approach for QoE-aware path selection outperformed state-of-the-art techniques approaches by reducing impaired videos in aggregate QoE by at least 37% and lowering accumulated video stall length four times. Based on the lessons learned with QoE prediction for traditional video streaming, we finally explore the VR video domain by introducing PERCEIVE and VR-EXP. PERCEIVE is a two-stage method for predicting the perceived quality of adaptive VR videos when streamed through mobile networks. By means of machine learning techniques, our approach is able to predict the playout performance of adaptive VR video and use this information to model and estimate QoE. In turn, VR-EXP consists of an experimentation platform that allows in-depth evaluation of state-of-the-art VR video optimization techniques. VR-EXP relies on software-based emulation to assess the interplay between a set of VR video optimization techniques and different levels of network performance.

Keywords: Quality of Service. Quality of Experience. Video Streaming. Virtual Reality. VR Video. Path Selection. Mobile Networks.

Seleção Escalável de Caminhos Sensíveis à QoE em Redes Definidas por Software

RESUMO

Desde o *streaming* de vídeo tradicional até os vídeos de Realidade Virtual (VR), a demanda por aplicativos de vídeo para dispositivos móveis está crescendo rapidamente. Para lidar com o tráfego massivo produzido por tais aplicativos, as operadoras de telefonia móvel contam com tecnologias de *offloading*, tais como *Small Cells*, *Content Delivery Networks* e, em breve, *Cloud Edge* e *5G Device to Device*. Embora essas técnicas sejam fundamentais para melhorar a eficiência da rede, elas produzem uma infinidade de caminhos pelos quais o tráfego do usuário pode ser encaminhado. No entanto, a expansão da diversidade de caminhos não fornece nenhuma garantia em relação ao incremento na Qualidade da Experiência (QoE) do usuário. Assim, surge um problema fundamental que consiste em como lidar com o crescente tráfego de vídeo enquanto se gerencia a interação entre otimização de infraestrutura e QoE. Resolver esse problema é notavelmente difícil, e investigações recentes não consideram o contexto de grande escala característico das redes de operadoras de telefonia móvel. Em suma, o problema do provisionamento dinâmico de caminhos sensíveis à QoE pode ser decomposto em duas funções fundamentais: (*i*) medição ou estimativa de QoE e (*ii*) provisionamento dinâmico de caminhos em uma rede programável. Para abordar o problema da estimativa de QoE, propomos um modelo para prever a qualidade do streaming de vídeo com base na observação dos indicadores de desempenho da rede IP subjacente. Para atingir esse objetivo, o modelo proposto utiliza medições ativas leves e técnicas de aprendizado de máquina. Em uma etapa adicional, introduzimos uma heurística inovadora para provisionamento de caminhos sensíveis à QoE em redes móveis de larga escala e baseadas em SDN. O método é baseado em dois componentes principais. O primeiro consiste em um algoritmo de tempo polinomial para compor múltiplas métricas de QoS. Já o segundo componente implementa uma estratégia escalável para tradução de QoS para QoE. Os resultados obtidos mostram que os métodos propostos para previsão de desempenho de streaming de vídeo produzem estimativas precisas. Como consequência, nossa abordagem para a seleção de caminhos sensíveis à QoE superou os métodos considerados como estado da arte ao reduzir a quantidade vídeos com QoE degradado em pelo menos 37 %, bem como diminuir o tempo de congelamento da reprodução de vídeo em quatro vezes. Com base nas lições aprendidas com a predição de QoE para streaming de vídeo tradicional, finalmente exploramos o domínio de vídeos VR introduzindo PERCEIVE e VR-EXP. PERCEIVE consiste em um método de dois estágios para predizer a qualidade percebida de vídeos VR adaptativos quando transportados por redes móveis. Por meio de técnicas de aprendizado de máquina, nossa abordagem é capaz de prever o desempenho de reprodução de vídeos VR e utilizar essas informações para modelar e estimar QoE. Por sua vez, VR-EXP consiste em uma plataforma de experimentação que permite uma avaliação detalhada de técnicas de otimização para vídeos VR. VR-EXP emprega emulação baseada em software para avaliar a interação entre um conjunto de técnicas de otimização e diferentes níveis de desempenho de rede.

Palavras-chave: Qualidade de Serviço. Qualidade de Experiência. Streaming de Vídeo. Realidade Virtual. Redes Móveis..

LIST OF ABBREVIATIONS AND ACRONYMS

ABR	Adaptive Bit Rate
AppQoS	Application Layer QoS
ARPU	Average Revenue Per User
AS	Autonomous System
BF-BW	Bellman-Ford - Bandwidth
CAPEX	Capital Expenditures
CDN	Content Delivery Network
CFA	Critical Feature Analytics
CP	Complexity Parameter
CSPF	Constrained Shortest Path First
D2D	Device-to-Device
DASH	Dynamic Adaptive Streaming over HTTP
DKS-Delay	Dijkstra - Delay
DPI	Deep Packet Inspection
FoV	Field of View
HAS	HTTP Adaptive Streaming
HMD	Head-Mounted Display
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ITU	International Telecommunication Union
LEAP	Lightweight AppQoS and QoE Predictor
LTE	Long-Term Evolution
MOS	Mean Opinion Score
MPD	Media Presentation Description

MPEG	Moving Picture Experts Group
MPLS	Multiprotocol Label Switching
MSE	Mean Squared Error
NBI	Northbound Interface
NCV	Nested Cross Validation
OPEX	Operational Expenditures
OWD	One Way Delay
PERCEIVE	Performance Estimation for VR Videos
PEVQ	Perceptual Evaluation of Video Quality
PLR	Packet Loss Rate
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QoS	Quality of Service
RMSE	Root-Mean-Square Error
SDN	Software-Defined Networking
SQAPE	Scalable QoE-aware Path Selection
SRD	Spatial Representation Descriptor
SWP	Shortest-Widest Path
TC	Traffic Control
TCP	Transmission Control Protocol
TE	Traffic Engineering
VR	Virtual Reality
VR-EXP	VR Video Experimentation Platform
XERROR	Cross-Validation Error

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

The future is mobile: wireless networks will account for more than two-thirds of all IP traffic by 2020 (ITU, 2017). In this context, operators are being challenged by video traffic, which is pushing their network infrastructure to the limit (MAALLAWI et al., 2015). According to Cisco, mobile video accounted for 60% of total Internet traffic in 2016. And there is more to come, since mobile video is expected to increase 9-fold by 2021, reaching 78% of total data traffic (CISCO, 2018). According to the same source, Virtual Reality (VR) videos will significantly increase this challenge as the traffic generated by this application is expected to increase 12-fold by 2022 (CISCO, 2018). One key enabler for supporting such a consistent growth is the diffusion of Head-mounted Displays (HMDs). HMDs are presenting high penetration rates as they (*i*) are becoming effective and affordable (*e.g.*, Google Cardboard/Daydream[1], (*ii*) are already provided at no cost with certain smartphones (*e.g.*, Google Pixel and Samsung Galaxy S series) and (*iii*) are being pushed by industry (*e.g.*, Facebook recently announced a permanent price drop in Oculus Go headset with the goal of reaching 1 billion VR users[2].

VR video streaming applications are challenging due to three main reasons: (*i*) they will run mostly over mobile networks, as mobile devices will account for 71% of the total IP Internet traffic by 2022 (CISCO, 2018); *ii*) mobile networks are characterized by highly variable levels of performance (FILHO et al., 2016); and (*iii*) VR video streaming applications demand high levels of network performance to achieve a satisfactory QoE (CISCO, 2018). To provide a notion of how demanding these applications are, recent studies have shown that, to provide adequate levels of QoE, current VR video applications require a network delay lower than 9 ms (FILHO et al., 2018), while the bandwidth needs for the upcoming ultra high definition VR will reach 500 Mbps (CISCO, 2018). At this level of demand, not only will network operators struggle to provide cost-effective services, but VR video content providers and developers will also be challenged by such resource-intensive applications.

To deal with the huge growth in data traffic, mobile operators have to constantly invest (*i.e.*, CAPEX and OPEX) to increase capacity, to switch technology (*e.g.*, 3G, 4G, 4G+, 5G), as well as to improve outdoor and indoor coverage. In the opposite direction, the Average Revenue Per User (ARPU) for mobile broadband has fallen from USD 23 in 2013 to USD 13 in 2015 (ITU, 2016). All those elements together place a lot of pressure

[1] Google VR: https://vr.google.com/

[2] A billion people in VR: https://goo.gl/2LNuAo

on operators to manage their infrastructure efficiently (MAALLAWI et al., 2015).

Aiming at increasing efficiency, mobile operators have been relying on offloading technologies such as Small Cells (Femtocell, Picocell), Wi-Fi offloading, Content Delivery Networks (CDNs), and, in the near future, 5G Device-to-Device communication (D2D) and 5G Mobile Edge Computing (HUQ et al., 2017; ANSARI et al., 2018; FRANGOUDIS; YALA; KSENTINI, 2017). These technologies are capable of offloading different segments of the network (*i.e.*, edge, aggregation, core and peering) and therefore play a fundamental role in network infrastructure optimization. Such approaches have the ability to shorten the distance between subscriber and content while avoiding network congestion by spreading the traffic among alternative paths. As an indication of how important these offloading techniques are, only in 2016, 60% of mobile data was relocated to alternative paths, just considering Wi-Fi and Femtocell offloading (CISCO, 2018).

1.1 Problem Statement

The adoption of offloading techniques introduces a multitude of possible paths through which user traffic can be forwarded and, as an immediate consequence, raises the complexity of the network management (*e.g.*, path selection, configuration and troubleshooting). While very important, such an advanced infrastructure does not directly translate into improved QoE (SCHLINKER et al., 2017). This is notably true if considering that some offloading techniques may rely on shared and third-party infrastructure, which would possibly exacerbate the unpredictability regarding the delivered QoE. Therefore, a challenging task for mobile operators consists in how to handle the increasing end-user traffic while optimizing infrastructure utilization and managing user's QoE. In fact, from the operator's perspective, addressing this challenge is crucial for improving competitiveness, since the effective management of the interplay between perceived QoE and infrastructure investments is the main factor for increasing the return of investment (NAM; KIM; SCHULZRINNE, 2016; AHMAD; FLORIS; ATZORI, 2016).

Given the context above, the main research challenge we intend to investigate is how to take advantage of the path diversity introduced by upcoming technologies (*e.g.*, 5G D2D, Edge Computing, and Fog Computing) to dynamically select paths capable of delivering cost-effective video applications. In a nutshell, the QoE-aware path selection task can be decomposed into two challenging problems. The first problem consists of timely predicting QoE for network paths. In turn, the second problem encompasses the

large-scale path selection algorithm, which should be able to take constraints (*e.g.*, target QoE and available network resources) into account and select optimized paths. The first problem is complex because the information that is closely related to QoE (*e.g.*, subjective evaluations and objective measurements) are not largely available or feasible to obtain in a systematic approach for large-scale networks. Regarding the second problem, it can be easily solved considering small deployments. However, it becomes notably complex when combined with additional constraints (*e.g.*, resource utilization) and applied to ultra-dense networks. In the following paragraphs, we provide an overview of the QoE-aware path selection problems examined in this Thesis.

QoE estimation for video streaming applications. Both the scientific community as well as the industry agree that maximizing the user's QoE regarding video streaming applications represents a central research challenge (KATSARAKIS et al., 2014). An essential aspect in this direction is to systematically determine the quality of the provided video services. To this end, service providers require a solution with low intrusion, scalability, and a reasonably accurate way to measure the quality of service delivered. This task becomes particularly challenging if encompassing cellular networks, in which highly intrusive measurement techniques would consume the resources available to subscribers. In an attempt to estimate QoE, most state-of-the-art approaches rely on QoS to QoE mapping functions since mobile operators already have tools in place to measure QoS metrics (JULURI; TAMARAPALLI; MEDHI, 2016).

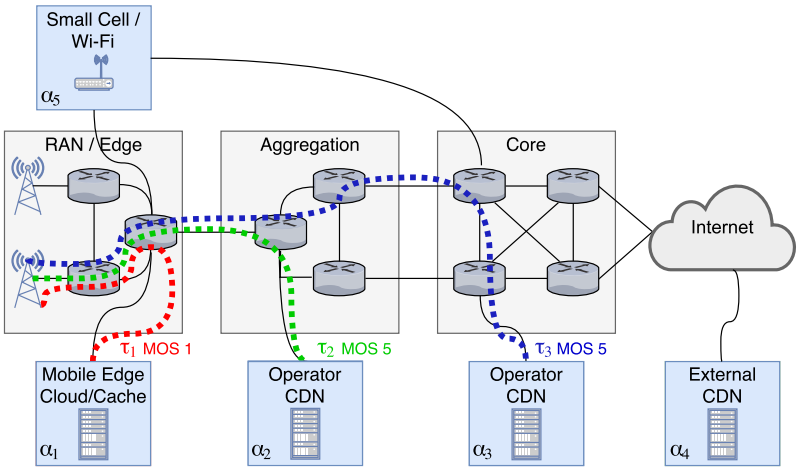
QoE-aware path selection and resource utilization minimization problem. To illustrate this problem, let us consider the simplified network topology shown in Figure 1.1. In the example, which is a temporal snapshot of a dynamic scenario, a subscriber requests a specific video (1080p - 4 Mbps of bitrate), which can be served by any of the three offloading containers (namely α_1, α_2 and α_3). Consider the paths τ_1, τ_2 and τ_3 have the following characteristics $\Phi(\tau) = \{$hop count, residual bandwidth[3] (Mbps), delay (ms), loss[4] (%)$\}$: $\Phi(\tau_1) = (3, 3, 5, 0)$, $\Phi(\tau_2) = (4, 10, 10, 0)$ and $\Phi(\tau_3) = (7, 100, 15, 0)$. Let us suppose the existence of an end-to-end QoS to QoE mapping function. In this case, one would use residual bandwidth, delay and packet loss, so it would be possible to estimate QoE for the paths τ_1, τ_2 and τ_3. Differently from τ_2 and τ_3, τ_1 does not provide the QoS performance needed to achieve the maximum QoE score (here we estimate QoE using the well-known Mean Opinion Score (MOS) ranging from 1 to 5) (ITU-T, 1998).

[3] In this document we consider residual bandwidth, TCP throughput and Bulk Transfer Capacity (BTC) as equivalent.

[4] In this document we consider delay and packet loss as one-way metrics.

Also, τ_2 should be preferred since τ_3 does not improve QoE and uses more resources (*i.e.*, number of links).

Figure 1.1: Mobile operator network and the increased path diversity introduced by offloading techniques



Source: by author (2019).

However, when dealing with large-scale dense networks, such an end-to-end approach leads to the evaluation of an immense number of paths, which would become impractical. One possible approach to solve this issue would be to consider paths as a composition of N links while using QoE estimation as link weight. However, the QoS metrics used to estimate QoE have different composition rules, which lead to an inconclusive aggregation result. In other words, the composition of N links with MOS 5 does not necessarily result in an MOS 5 path (WANG; CROWCROFT, 1996). Another alternative would be to compose the multiple QoS metrics along the path and then apply the MOS estimation when the path is finally composed. However, the composition of one concave, one additive and one multiplicative metric was shown to be NP-Complete (WANG; CROWCROFT, 1996).

Finally, it would be possible to employ one of the following strategies: (*i*) use a single QoS metric composition (*e.g.*, residual bandwidth), since it is the most influential when predicting QoE (CASAS et al., 2016), or (*ii*) use delay, which is an excellent metric for sensing queue occupation (WANG; CROWCROFT, 1996) or, finally, (*iii*) a combination of both these metrics used in conjunction with a modified version of Dijkstra, as it has shown to be a special case of aforementioned NP-Complete problem, which has polynomial solution, named Shortest-Widest Path (SWP) (WANG; CROWCROFT, 1996). Considering that those approaches are effective in avoiding bottlenecks, they would consequently improve QoE. However, from a mobile operator's perspective, these path selec-

tion approaches fail at a critical point: they perform optimal decisions only within a time frame and do not take the long-term operation (*i.e.* infrastructure resource utilization) into consideration.

1.2 Hypothesis

The hypothesis we considered as the fundamental research challenge to guide this Ph.D. research work is the following:

QoE-aware path selection strategies contribute to cost-effective video streaming.

The objective of this thesis is to verify the hypothesis above by answering the following research questions.

Research Question 1. Considering that mobile operators already have tools in place to measure network performance, would it be possible to employ monitored network indicators to predict playout performance and QoE for both traditional 2D and VR video streaming applications?

Research Question 2. How to employ QoE prediction to dynamically select and deploy paths that maximize QoE and minimize infrastructure utilization over time?

1.3 Goals, Contributions and Scope

The Thesis has five main goals: (*i*) devise a prediction model capable of estimating video streaming playout performance and QoE based on available network information; (*ii*) formalize the QoE-aware path selection problem; (*iii*) formulate a QoE-aware path selection heuristic with the ability to operate in dense networks; (*iv*) propose a QoE prediction model for VR videos; and (*v*) provide an in-depth evaluation of state-of-the-art VR video optimization techniques. These goals unfold into five main research contributions, as enumerated next.

First, we introduce a model which relies on active IP performance measurements to predict video quality. In a first stage, the model takes as input the observation of performance indicators of the underlying IP network (*i.e.*, delay, loss and residual bandwidth) (FILHO et al., 2016). Next, it explores regression decision trees in order to estimate both the video playout performance and QoE.

The second contribution encompasses the formalization of the QoE-aware path selection problem.

As for the third contribution, we address the QoE-aware path selection problem by proposing a polynomial time complexity path selection heuristic. This algorithm takes advantage of the previously proposed QoE prediction model and introduces a novel algorithm for performing efficient QoE-aware path selection, in Software-defined Networks, based on per-link QoE composition. The proposed method has the ability to perform dynamic path selection by providing a feasible balance between QoE maximization and network resource minimization over time.

Bringing the QoE prediction model to the virtual reality arena, as the fourth contribution we introduce PERCEIVE, a two-stage adaptive VR performance assessment model. It employs regression decision trees to predict VR video playout performance using network QoS indicators as predictors. Then, it uses the video playout performance metrics to model and estimate the end-user perceived quality.

When considering both the multitude of approaches to optimize a VR video streaming and the highly variable mobile network performance, it becomes a difficult challenge to understand how different (combinations of) optimization techniques perform under varying infrastructure conditions. To fill in this gap, as the fifth contribution we introduce VR-EXP, an adaptive VR video streaming experimentation platform. The platform is capable of systematically evaluating different combinations of VR video streaming optimization approaches. Also, VR-EXP allows pinpointing the interplay between a set of optimization techniques and variable network performance.

In summary, in this thesis we are interested in evaluating the efficiency of a dynamic path selection algorithm that benefits from accurate QoE prediction models. Thus, we focus on elements of the video streaming domain that are affected by the network performance (*e.g.*, video fetching method, adaptive bitrate heuristic and playout buffer dimensioning). Aspects such as video CODEC, saliency detection and video rendering method are outside the scope of this work.

1.4 Organization

The remainder of this document is organized as follows.

- In Chapter 2, we present state-of-the-art approaches regarding QoE estimation and

QoE-aware path selection.

- In Chapter 3, we introduce LEAP, a model for predicting video playout performance and QoE based on network performance indicators.
- In Chapter 4, we formally define the path selection problem and propose a scalable QoE-aware path selection scheme for SDN-enabled mobile networks.
- In Chapter 5, we describe PERCEIVE, a two-stage method for predicting the perceived quality of adaptive VR videos when streamed through mobile networks.
- In Chapter 6, we introduce VR-EXP, a novel experimentation platform for VR video streaming.
- In Chapter 7, we present our conclusions along with a perspective for future work.

2 BACKGROUND AND STATE OF THE ART

In this chapter, we explore fundamental concepts as well as the most prominent investigations related to QoE-aware path selection for video streaming applications. We start by presenting the main elements of traditional video streaming. Next, we discuss relevant aspects of QoE estimation for traditional video streaming. Then, we examine state-of-the-art QoE-aware path selection schemes. Next, we explore cutting-edge techniques for the VR video streaming ecosystem, namely spherical-to-plane projection, CODECS, viewport-aware rate adaptation heuristics and prediction error for VR video streaming. We conclude this chapter by reviewing important research work concerning adaptive bitrate algorithms and buffer management for VR video streaming.

2.1 Fundamental Concepts for Video Streaming

Apple HTTP Live Streaming (HLS) and MPEG-DASH (SODAGAR, 2011) are two examples of HTTP Adaptive Streaming (HAS) implementations. MPEG-DASH is an open standard which has been widely adopted by video clients. Basically, the MPEG-DASH approach is focused on encoding the video content at multiple quality representations (*i.e.*, bitrates), while each quality representation is time-segmented into small parts (*i.e.*, segments). These segments are described in a Media Presentation Description (MPD) file, which contains information such as segment duration, available bitrates, and timing. Then, during the streaming session, the video client will perform the adaptive bitrate streaming. To do so, the video client relies on network bandwidth estimations to decide, for each video segment, the quality representation that best fits the available bandwidth. Although the DASH standard provides a comprehensive specification of the video streaming service, it is agnostic to some important components such as the Adaptive Bitrate (ABR) logic and the video CODEC.

One critical component of the video client is the adaptive bitrate streaming. ABR algorithms are complex because they must manage the available bandwidth while maximizing the quality representation and minimizing the stall probability. Although the ABR algorithms for traditional 2D video streaming have been extensively studied, recent investigations (AKHTAR et al., 2018; SPITERI; SITARAMAN; SPARACIO, 2018) have shown that this is still an open-source research problem. As an example, it has been demonstrated the possibility to significantly improve the performance of state-of-the-art

ABR algorithms, namely BOLA (SPITERI; URGAONKAR; SITARAMAN, 2016) and MPC (YIN et al., 2015a). Akhtar *et al.* (AKHTAR et al., 2018) found that both BOLA and MPC algorithms rely on parameters that are sensitive to variable network performance, so they may perform poorly under certain conditions. To fill this gap, the authors introduce VirtualPlayer (AKHTAR et al., 2018), a trace-based simulator that mimics the behavior of a traditional video streaming player. It allows, for example, to investigate ABR algorithms when subjected to real-world networks. In the same direction, Spiteri *et al.* (SPITERI; SITARAMAN; SPARACIO, 2018) introduce Sabre, an open-source simulation tool that enables simulating ABR algorithms for 2D videos when subjected to real-world requirements.

2.2 QoE Estimation

One possible path toward estimating QoE for video applications is to interview users asking for their opinion. Alternative approaches rely on objective video quality measurements which, under certain circumstances, can approximate the users' QoE. When considering the video streaming domain, Mean Opinion Score (MOS) (ITU-T, 1998) is commonly employed to capture the subjective user opinion regarding a video session. MOS is expressed on a numerical scale which ranges from 1 to 5, where 1 means a poor experience and 5 represents an excellent quality of experience. Typically, the score is assigned by the user after the video streaming session ends. Alternatively, QoE can be estimated by objective evaluation techniques. Perceptual Evaluation of Video Quality (PEVQ) (CHIKKERUR et al., 2011), Structural Similarity (SSIM) (WANG et al., 2004) and Peak Signal-to-Noise Ratio (PSNR) (WINKLER; MOHANDAS, 2008) are examples of techniques that rely on picture quality models to provide a video quality estimation. At a high level of abstraction, these methods perform a frame-by-frame comparison between the original video and the streamed one. As these methods require the original video to perform the quality assessment, they are referred to as Full Reference (FR) methods.

Traditional mechanisms such as MOS, PSNR, PEVQ and SSIM are not scalable when applied to systematically evaluate if a network infrastructure provides the required quality to support video streaming applications. This is especially true when considering the high costs involved with user interviews and the overhead involved in full reference techniques. Msakni and Youssef (MSAKNI; YOUSSEF, 2013) analyzed several techniques for QoE prediction which do not rely on video transfer. They concluded that none

of the considered approaches could be deemed reliable. Essentially, the non-linearity of human opinion compromises accuracy when using network parameters as direct predictors of QoE, since a given configuration can be graded differently in MOS. An alternative strategy would be to evaluate video quality by objectively grading the user's opinion. However, techniques that allow for such a measurement require the download and analysis of real video files, entailing a substantial increment in network traffic. In an attempt to overcome these limitations, recent investigations proposed to estimate quality using alternative data acquisition procedures, compatible with large-scale scenarios. The most significative advances in this regard are summarized below.

A first group of investigations is characterized by techniques that estimate QoE indicators using network-based information. Due to not taking into account application performance, these techniques do not allow an accurate understanding of the application behavior (PESSEMIER et al., 2013; XIAO et al., 2015). As a representative example of research work in this group, De Pessemier *et al.* (PESSEMIER et al., 2013) evaluated the influence of network QoS parameters on the quality of user experience when using video applications with DASH. The authors carried out experiments in which network QoS indicators, such as delay and throughput, have been degraded in a controlled manner. At the same time, the user opinion regarding the quality of video playback was measured using MOS.

In another significant work in this group, Xiao *et al.* (XIAO et al., 2015) proposed a model to estimate the user opinion based on scheduling algorithms running at a base station. In this case, the authors correlated offline trace files in order to capture the relationship between the scheduling algorithm employed by the Node B and the respective QoE (informed by the end user through an interview process). Despite minimizing the scalability problem by adopting a prediction model, the proposed method is not appropriate for service providers because the same network performance can be mapped into different MOS scores (due to the subjectivity of the human opinion).

The second group of related work tries to establish the relationship between the application performance (measured in the video streaming client) and QoE, without considering network QoS indicators. Balachandran *et al.* (BALACHANDRAN et al., 2012) tried to evaluate the impact of those effects on user experience. To measure the performance of video playback, three indicators were proposed: startup delay, stall count and total stall length. Along these lines, Balachandran *et al.* (BALACHANDRAN et al., 2013) introduced a QoE prediction approach that takes as input performance indicators

measured at the application layer. To this end, the authors considered a database of performance indicators obtained directly from user's clients during video sessions. Even though the resulting predictions are valuable for the provider to estimate QoE, this information is incomplete since it does not allow the provider to understand the influence of network performance on user experience. The lack of connection among application layer performance, QoE and network performance hinders the service provider's ability to understand the network influence on QoE indicators.

2.3 QoE-aware Path Selection

This section is organized around four main groups of investigations. The first group is characterized by techniques that perform path selection considering, among other elements, QoE indicators measured at end-user devices. QoE indicators for video streaming such as startup time, amount/duration of stalls and buffer events, when measured at end-user devices, present high accuracy, since they are obtained as close to the user as possible (BALACHANDRAN et al., 2013; RAMAKRISHNAN et al., 2015; NAM et al., 2014). However, these techniques entail lower system scalability, reduced flexibility and the requirement of end-user collaboration through the installation of measurement software on their devices (LIOTOU et al., 2015). Additionally, constraints such as privacy policies, reduced battery life and the end-users themselves, who are usually unwilling to install additional software on their device, make this group of solutions less attractive for mobile operators.

Considering the issues involved in employing end-user data, a second group of the related work makes use of server-side information, available at the content provider equipment, to estimate QoE and perform path selection (SCHLINKER et al., 2017; UZAKGIDER; CETINKAYA; SAYIT, 2015; GANGWAL et al., 2016). From the mobile operator's perspective, obtaining server-side data from several video streaming providers can be a challenging task, hindering the capacity of an operator to provision paths autonomously. Additionally, neither QoE information available at server-side nor at client-side allows the operator to isolate the influence of each link on the QoE score.

The third group of related work relies on network-side information in order to estimate the quality of experience and select QoE-aware paths (ECKERT; KNOLL; SCHLEGEL, 2013; FARSHAD et al., 2015). A representative research work in this group (ECKERT; KNOLL; SCHLEGEL, 2013) leverages Deep Packet Inspection (DPI)

to analyze video streaming flows and predict both the video playout performance and user experience. The drawback of this approach relies on the fact that deep packet inspection demands a high amount of resources besides being complex to deploy in a large network.

Trying to overcome the limitation of high resource consumption, in another significant work in this group, Farshad *et al.* (FARSHAD et al., 2015) proposed a lightweight approach. The proposed framework takes advantage of SDN to select and replicate specific segments of video streaming flows. In a second step, this information is forwarded to an analysis subsystem. One key aspect of this approach consists in analyzing just partial information, as the manifest files of each video streaming request, instead of the whole flow stream. Once the analysis is complete, the proposed approach is able to estimate QoE for individual video sessions considering only network-side information.

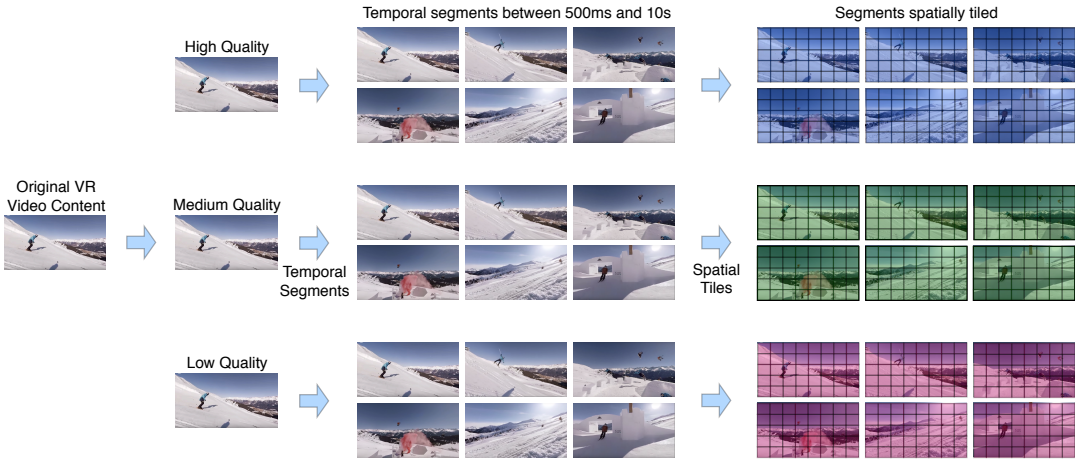
The main limitation of methods that rely on observation of video streaming data traffic, inside the network, is related to the proliferation of the HTTPS protocol. Important players such as Netflix and Youtube have adopted HTTPS as the standard protocol for video streaming delivery. Recent research work has stated that, given the widespread adoption of HTTPS for video delivery, approaches based on streaming flow analysis became unpractical (CASAS et al., 2016).

Finally, we consider the fourth group of investigations concerning QoS-aware path selection (ZHANG; HAO; MOUFTAH, 2014; KUIPERS et al., 2002; WANG; CROWCROFT, 1996). These techniques do not directly consider QoE, which may lead to an overemphasis on a single (or a couple of) QoS metric. Variations of the Dijkstra or Bellman-Ford algorithms that consider only QoS either use unnecessarily long paths (when oriented to maximize residual bandwidth) or do not appropriately consider paths with impaired quality (when oriented to shortest paths).

2.4 Spherical-to-Plane Projection and CODECS

Figure 2.1 provides an overview of the VR video encoding process. The first step consists in encoding different quality representations (*e.g.*, high, medium and low quality) from the original VR video content (RAW file). Next, each quality representation is split into segments. Each segment contains a temporal fraction of the video (*e.g.*, from 500 ms to 10 s), according to the HAS scheme. Finally, each segment is spatially split into an specific tiling scheme (*e.g.*, 8x5 - 8 horizontal by 5 vertical tiles) by using a modern video encoder (*e.g.*, HEVC/H.265).

Figure 2.1: Overview of the VR encoding process.



Source: by author (2019).

One effective strategy to reduce the huge bandwidth demands of 360-videos is delivering only the viewport in high resolution, streaming the remaining area of the video in low resolution or not streaming it at all. To achieve this spatial segmentation of the panoramic view, several approaches explore spherical-to-plane projection techniques (GRAF; TIMMERER; MUELLER, 2017; CHEN; LI; ZHANG, 2018; CORBILLON et al., 2017; HRISTOVA et al., 2018; HOSSEINI; SWAMINATHAN, 2016; ZHOU et al., 2018). In a recent investigation, Graf *et al.* (GRAF; TIMMERER; MUELLER, 2017) examine the bitrate overhead and bandwidth requirements of distinct tiling schemes (*i.e.*, 1x1, 3x2, 5x3, 6x4 and 8x5) implemented within modern video codecs (*e.g.*, HEVC/H.265 and VP9). By applying Peak Signal-to-Noise Ratio (PSNR) within the VR video viewport, the authors assessed the video quality and concluded that the 6x4 tiling scheme provides the best trade-off among viewport selection flexibility, bitrate overhead, and bandwidth requirements. In a similar direction, Zhou *et al.* (ZHOU et al., 2018) further examine this field by comparing standard spherical projection approaches to offset projection techniques. The latter are characterized by distorting the spherical surface to allow the convergence of the pixels of the VR video in a particular direction. Offset projections are significantly more complex than traditional projection techniques because they demand a simultaneous control of bitrate and view orientation adaptations. By employing PSNR and Structural Similarity (SSIM), the authors concluded that, in general, offset projections can provide better quality than their non-offset counterparts. Despite their contributions, the conclusions of these investigations are limited because they do not consider important variables, such as the effects of variable viewport prediction error and parallel fetching methods (such as HTTP/2) on their approaches. Also,

the considered approaches were evaluated considering limited or unrealistic network performance conditions.

In another important investigation, Chen *et al.* (CHEN; LI; ZHANG, 2018) analyze recent advances regarding alternative projection methods, including viewport-dependent and viewport-independent approaches. The central objective of this work is to assess both the coding efficiency and distortion introduced by each approach. Besides valuable quantitative and qualitative insights regarding a wide range of projection schemes, the authors conclude that in order to effectively evaluate such a wide range of projection schemes, a more sophisticated evaluation process is required. The main reason for this conclusion is that traditional PSNR computes the whole projection map, which cannot handle viewport-dependent projections. Additionally, due to the unpredictability of viewport prediction errors, the areas surrounding the viewport should also be considered in the quality evaluation, but with a reduced weight. In this investigation, the authors also review alternative metrics for video quality assessment proposed by JVET (BOYCE et al., 2017). They conclude that although several flaws of conventional PSNR have been fixed, a more comprehensive method for evaluating video quality for viewport-dependent VR videos is still missing.

2.5 Viewport-aware VR Video Streaming

In order to provide an immersive user experience, VR videos demand significantly higher bandwidths when compared to traditional video streaming. These ultra-high bandwidths are not always available in wireless networks and are not easy to process by lightweight mobile devices. In fact, currently, the streaming of VR videos through mobile networks is far from optimal. A VR video contains a full 360° panoramic view, regardless of the fact that only a fraction of it, namely the viewport[1], is visible at any given instant. In an attempt to optimize bandwidth usage, a recent research path has proposed viewport-aware schemes for VR video streaming, based on HAS variants in which quality representations are not only segmented in time but also spatially split into smaller pieces (*i.e.*, tiles) (CONCOLATO et al., 2017).

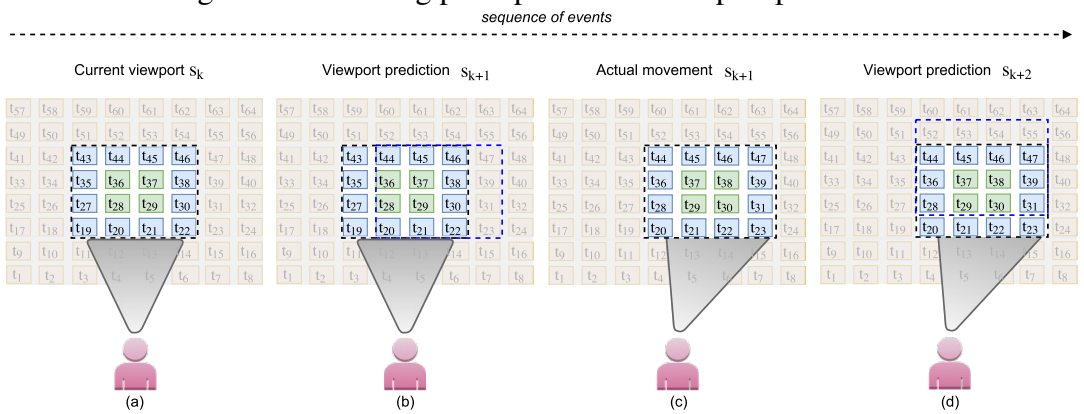
An important aspect to consider in adaptive tile-based VR streaming services is viewport prediction, which allows optimizing bandwidth usage considerably. Since a full VR video can easily reach 12K video resolution (CORBILLON et al., 2017), most video

[1] Also referred to as Field of View (FoV)

players rely on heuristic algorithms to predict near-future user's head movements. By considering next position prediction, the video player is able to request only tiles that are likely to be inside the viewport, which leads to reduced bandwidth utilization. To provide this prediction, heuristic algorithms consider variables such as the angular velocity of the user's head, movement patterns for previous viewers, video content (*e.g.*, in a football match users will most likely follow the ball's movements), among other factors (BAO et al., 2016). With such predictions, the video player can reduce bandwidth utilization in up to 72% (HOSSEINI; SWAMINATHAN, 2016).

In practice, the viewport prediction algorithm is responsible for keeping a small video playout buffer (*e.g.*, 2 seconds) with the tiles that are more likely to belong to the viewport in the future. To illustrate how the viewport prediction interacts with the playout buffer, consider the example of a user watching a tile-based VR video using an HMD. Assume a given temporal segment S_k and a respective viewport V_k, as depicted in Figure 2.2 (a). At this moment, the video player is requesting high-resolution chunks only for tiles inside the viewport V_k. Then, based on the near-future viewport prediction for the next segment (S_{k+1}), the video player starts requesting high-resolution tiles for the viewport V_{k+1} (delimited by the right dashed square in Figure 2.2 (b)). As predicted, the viewer slightly moves to the right (see Figure 2.2 (c)). At this point, driven by the viewport predictor, the VR player starts requesting high-resolution chunks for the predicted viewport on the segment S_{k+2} (Figure 2.2 (d)), and so forth.

Figure 2.2: Working principles of the viewport prediction.



Source: by author (2019).

To perform the viewport prediction, most approaches follow a similar procedure, which includes processing one or more input information, applying a prediction method, and then checking the prediction accuracy. As input, prediction algorithms can rely on past users' head motion (HOU et al., 2018; QIAN et al., 2016), fixation point ac-

celeration (NGUYEN; YUN, 2018), fixation point angular velocity (NGUYEN; YUN, 2018; PETRANGELI et al., 2017; FAN et al., 2017), image saliency maps and motion maps (FAN et al., 2017), or even sound localization information (JEONG et al., 2018). In turn, to perform the viewport prediction itself, state-of-the-art approaches usually rely on deep learning (HOU et al., 2018), mathematical modeling (NGUYEN; YUN, 2018; PETRANGELI et al., 2017; JEONG et al., 2018; QIAN et al., 2016), or neural networks (FAN et al., 2017). Finally, the prediction accuracy is assessed by subjecting the prediction model to traces containing realistic head-tracking information (*i.e.*, ground truth). Thus, the residual error can be evaluated. By performing such predictions, the VR video player can, according to He *et al.* (HOSSEINI; SWAMINATHAN, 2016), reduce bandwidth utilization in up to 72%.

As discussed, viewport prediction is a sensitive task, which might affect the user's perception in unexpected ways. Errors on the prediction of the viewport (*i.e.* the Field of View (FoV) that the user will look at in the next segment) may lead to partial or full degradation of the perception, even if the network conditions are enough to guarantee the user's QoE. This means that, during the streaming, two different processes (namely the viewport prediction and the effects of the network on the adaptive streaming performance metrics) will have a major influence on the user's QoE.

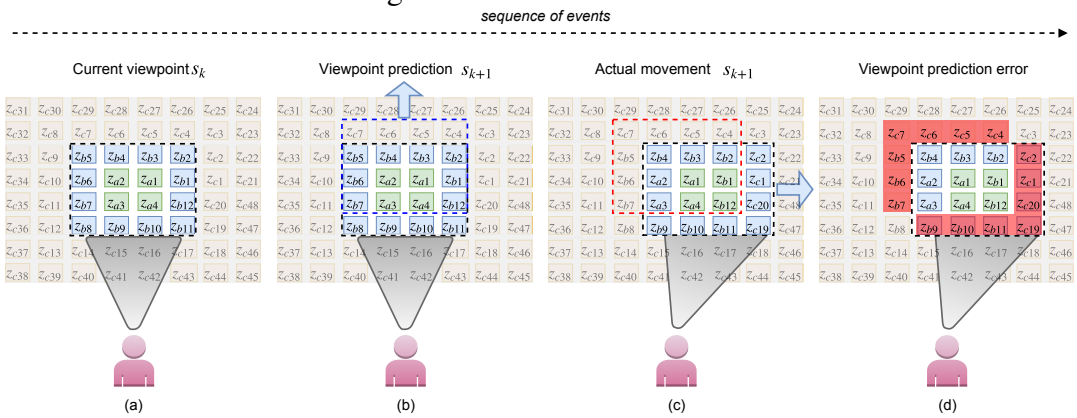
2.6 Viewport Prediction Error

Prediction errors are very likely to occur due to the randomness of users behavior. Besides, prediction algorithms may considerably decrease their accuracy when the playout buffer is increased. For example, the prediction accuracy can drop from 90% to approximately 60% if the prediction window is increased from 1 to 2 seconds (QIAN et al., 2016). However, an increased playout buffer may be crucial to operate in current mobile networks, which are characterized by highly variable performance conditions, even in short time frames. Considering these intricacies, an effective assessment of viewport prediction algorithms should consider (and quantify) how the error rate of a particular algorithm affects QoE when combined with other optimizations (*e.g.*, buffer management heuristics and dynamic rate adaptation algorithm) and subjected to realistic network performance conditions.

To illustrate how the viewport prediction error occurs, consider the example of a user watching a tile-based VR video using a head-mounted display. Consider a given

temporal segment S_k and a respective viewport V_k, as depicted in Figure 2.3 (a). At this moment, the video player is requesting high-resolution chunks only for tiles inside the viewport V_k. Then, based on the viewport prediction for the next segment (S_{k+1}), the video player starts requesting high-resolution tiles for the predicted viewport V_{k+1} (delimited by the blue dashed square in Figure 2.3 (b)). However, rather than moving his/her head up, consider that the viewer actually slightly moves to the right (see Figure 2.3 (c)). At this point, due to the viewport predictor error, the VR player requested seven tiles in high-resolution which will not actually be displayed (upper left red tiles in Figure 2.3 (d)). Likewise, seven low-resolution tiles turned out to belong to the viewport (bottom right red tiles in Figure 2.3 (d)). As one can observe, viewport prediction is a sensitive task. Viewport prediction errors may lead to partial or full degradation of the perceived quality, even if the network performance conditions are enough to guarantee the user's QoE.

Figure 2.3: Prediction error.



Source: by author (2019).

2.7 Adaptive Bitrate Algorithms and Buffer Management for VR Video Streaming

Taking viewport prediction information as input, most approaches rely on per tile rate adaptation algorithms. This method allows reducing the amount of information to be downloaded by keeping only the viewport's tiles in high resolution. State-of-the-art approaches for adaptive bitrate in VR videos differ from each other mainly with respect to how they manage the balance between video quality and available bandwidth while considering the spatial segmentation. For example, Petrangeli *et al.* (PETRANGELI et al., 2017) consider a multi-zone VR video and propose a per tile quality selection heuristic.

The algorithm starts by selecting the highest available quality for the inner tiles (close to the fixation point), and then repeats this procedure for the outer zones until the residual bandwidth is exhausted. This approach alleviates the edge effect (transition between different quality representations). Thus, it provides superior VR video quality at the cost of increased bandwidth consumption.

He *et al.* (HE et al., 2018) propose to simultaneously optimize, among other parameters, playout bitrate and buffer occupancy. Similarly to Petrangeli *et al.* (PETRANGELI et al., 2017), they perform bitrate adaptations depending on the position of each tile concerning the current fixation point. However, they introduce a learning strategy with the ability to avoid performance degradation for future segments by automatically adapting the buffer reservation. By using a fine-grained bitrate adaptation, these investigations were able to reduce the bandwidth utilization in 35% and 40%, respectively.

Graf *et al.* (GRAF; TIMMERER; MUELLER, 2017) advances a step forward in the state of the art by providing a comprehensive investigation with respect to essential components of the VR ecosystem. The authors introduce three tile scheme strategies for ABR, namely Full Delivery Basic, Full Delivery Advanced and Partial Delivery. These schemes drive the ABR algorithm regarding the bitrate adaptation for both the viewport and the remaining tiles. For example, in Full Delivery Basic scheme, all the tiles belonging to the viewport are requested in the highest available quality, while the remaining tiles are requested in the lowest quality, regardless of the available bandwidth. The Partial Delivery scheme employs an aggressive bandwidth saving strategy, requesting only the tiles within the viewport in high resolution, while the remaining tiles are not requested at all. The authors evaluated several projection schemes (as discussed in Subsection 2.4), combined with multiple segment sizes. By assessing the bitrate overhead, bandwidth requirements, and viewport quality, this approach achieved bandwidth savings from 40% to up to 65% when compared to state-of-the-art techniques.

Closely related to ABR algorithms, the playout buffer management plays a vital role in the VR video realm. As discussed earlier, an increased playout buffer size consists in an effective way to protect against stalls (*i.e.*, empty buffer) caused by network performance fluctuations. On the other hand, a small playout buffer is necessary to keep the accuracy of viewport prediction methods within acceptable levels. Specifically on this subject, Ma *et al.* (MA et al., 2018) propose a dynamic buffer size management method which is guided by a constrained optimization model. This method aims at maximizing

QoE by adjusting the buffer size based on the viewport prediction error and available bandwidth. Throughout simulation experiments, the authors claim gains from 2.7% up to 6.7%, in terms of QoE, when compared to non-dynamic buffer size approaches. In another relevant investigation, Almquist *et al.* (ALMQUIST et al., 2018) present a data-driven study which explores the trade-off between the playout buffer size (*i.e.*, prefetching aggressiveness) and viewport prediction errors. The prefetching aggressiveness is evaluated while considering different VR video categories (*i.e.*, exploration, static, moving, rides and misc.). The authors provide valuable qualitative and quantitative insights regarding how to best address the prefetching aggressiveness trade-off. As a key insight, they demonstrate that the accuracy of the prediction varies significantly among different categories. Additionally, in line with previous investigations, they found that adequate levels of viewport prediction accuracy are observed only within a very small time frame.

The aspects discussed above evidence that the VR video ecosystem is substantially more complex than traditional video streaming. To the best of our knowledge, neither a QoE evaluation model nor a QoE prediction method is available in current VR environments. After concluding the state of the art analysis, in the next section, we present a model for performance prediction of video streaming applications, which is the first building block of the proposed QoE-aware path selection scheme.

3 PERFORMANCE PREDICTION FOR VIDEO STREAMING APPLICATIONS

In this chapter, we introduce a QoE prediction model for video streaming based on indicators obtained from the underlying IP network[1] (FILHO et al., 2016). The proposed model explores decision trees in order to determine the relation between network QoS indicators[2] and objective indicators representative of the video reproduction quality experienced by the end user, henceforth designated as AppQoS[3]. Additionally, by means of AppQoS processing, the model allows the inference of user QoE.

The remainder of this chapter is organized as follows. Section 3.1 presents the proposed prediction model. Section 3.2 details the experiment configuration and model construction aspects. Section 3.3 reports performance evaluation as well as potential model applications.

3.1 LEAP: Lightweight AppQoS and QoE Predictor

As just mentioned, we propose a Lightweight AppQoS and QoE Predictor (LEAP). It is capable of providing a detailed view of how the network performance affects video streaming applications and, moreover, the corresponding user experience. Figure 3.1 shows the general model scheme. The model is designed to receive four network performance indicators as input: (*i*) delay, (*ii*) jitter, (*iii*) throughput and (*iv*) packet loss. To capture video playback performance, the model predicts three video playout performance indicators: (*i*) startup time, (*ii*) stall count and (*iii*) total stall length. To estimate each AppQoS indicator, the four network QoS indicators are analyzed together. In a second stage, the three AppQoS indicators are used to estimate QoE (using the MOS score).

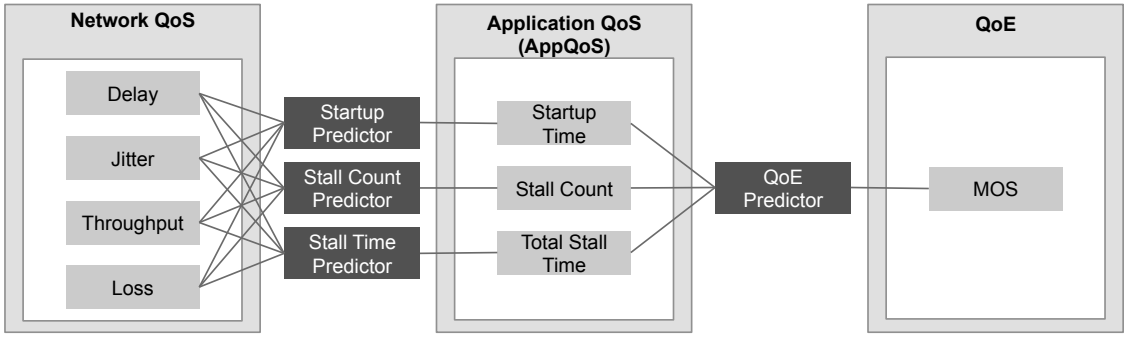
[1] This chapter is based on the following publication:

- Roberto Iraja Tavares da Costa Filho, William Lautenschlager, Nicolas Kagami, Valter Roesler, Luciano Paschoal Gaspary. **Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE**. IEEE Global Communications Conference (GLOBECOM 2016).

[2] The term network QoS indicators refers to performance of IP networks.

[3] The term AppQoS refers to the last objective barrier capable of being measured in the context of the end user.

Figure 3.1: LEAP general scheme



Source: by author (2019).

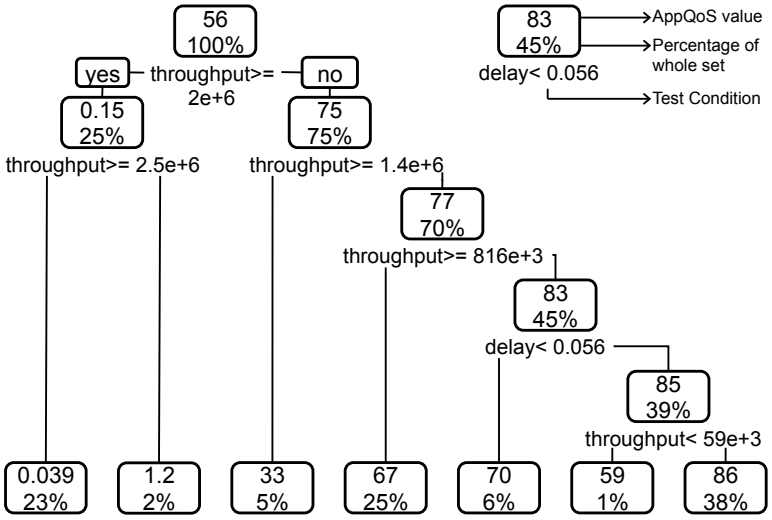
3.1.1 AppQoS Prediction

LEAP leverages the Regression Decision Tree technique to construct the three AppQoS prediction models. It was chosen mainly due to its suitability to handle complex and non-linear relationships between attributes. Additionally, decision trees have three characteristics that are desirable in the network management realm. First, the resulting prediction structure is simple to implement and integrate with third-party network management systems (they require only decision structures). Second, the prediction scheme presents low time complexity, being suitable for large scale environments. Third, decision trees provide decent accuracy. Although a more sophisticated approach for machine learning (*e.g.*, deep learning) might present higher accuracy than decision trees, for this task, the accuracy provided by decision trees is good enough. More accurate algorithms would not justify increased complexity for implementation and integration.

During the training stage, the model learns how each attribute X affects the response variable Y. In our case, the network QoS indicators are the X attributes and the video streaming performance indicators (AppQoS), Y. Once the training stage is finished, LEAP is able to estimate AppQoS indicators by comparing the measured QoS indicators against the decision trees. The model evaluates each QoS indicator, node by node, until it reaches a leaf, where the predicted AppQoS is defined. For example, in Figures 3.2 and 3.3 it is possible to observe a portion of the decision trees for total stall length and stall count for 1080p videos. To improve legibility, we present two partial excerpts from the resulting decision trees. Their constructive aspects and a performance evaluation will be presented in Subsection 3.2.3.

Each decision tree is built using a recursive procedure that executes successive splits, starting from a single node containing all attributes (QoS parameters). This proce-

Figure 3.2: Decision tree for total stall length of 1080p video playout



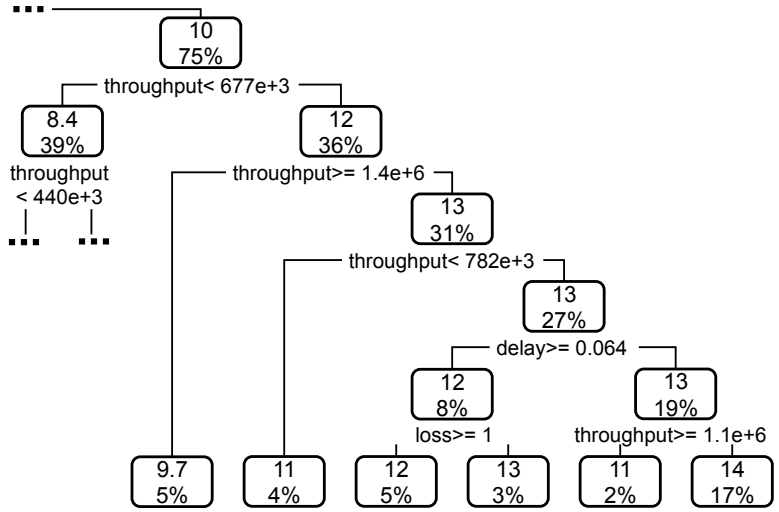
Source: by author (2019).

dure defines the tree growth and is controlled by two parameters. The first is the *minsplit*, which represents the minimum number of related observations for a new branch to be created. The second parameter is called *cp*, which describes the minimum gain, regarding error reduction, that a specific node needs to provide in order to be created. Each resulting decision tree has a particular cross-validation error (X), which provides an estimated error when testing the prediction structure against independent data (a portion of the dataset not used during the training stage). Finally, in the pruning stage, each pair of leaves with a common parent is tested for merge according to the Mean Squared Error (MSE). If MSE is reduced, then those leaves are removed and their parent becomes a leaf node. Otherwise, the structure remains the same for those nodes.

$$MSE = \sum_{c \in leaves(T)} \sum_{i \in c} (y_i - m_c)^2 \tag{3.1}$$

Correct parametrization is a key factor when aiming at building an efficient estimation mechanism. Figures 3.4, 3.5 and 3.6 illustrate three important parameters to construct the decision tree for startup time prediction of 1080p videos. In Figures 3.4 and 3.5, it is possible to observe that both the cross-validation error and the R^2 stabilize after five splits. Thus, one can conclude that five branches is the best depth for this structure. Growing the tree beyond this point will not help to lower the error or increase the data fit. In Figure 3.6, it is possible to observe the heat map that shows the resulting interpolation matrix between *minsplit* and *cp*, where the darker areas represent lower cross-validation error values. In this case, the optimal parameterization for the startup time decision tree

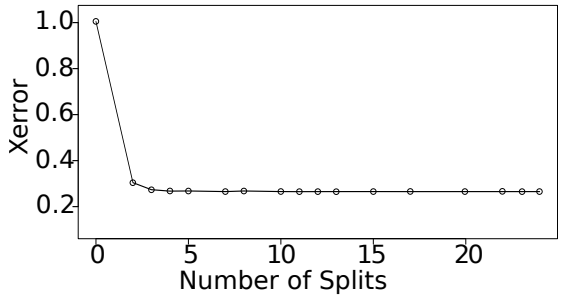
Figure 3.3: Decision tree for stall count of 1080p video playout



Source: by author (2019).

was *minsplit*=24, *cp*=10^{-22} and *splits*=5.

Figure 3.4: LEAP parameterization: Xerror VS Split



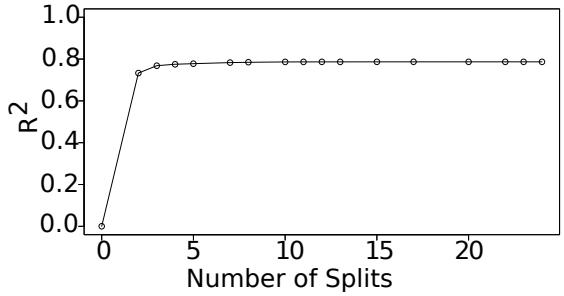
Source: by author (2019).

Finally, it is important to emphasize that LEAP is divided into two distinct phases. In the first stage, the tasks with high complexity, such as construction and training of the LEAP model, are performed offline. In particular, the construction of an optimal decision tree is known to be an NP-Complete problem, so a heuristic algorithm is used to obtain a near optimal structure. In turn, the second stage occurs as an online procedure. Whenever QoS indicators are available in the database, a subroutine estimates the AppQoS and QoE indicators just by comparing them against the decision tree thresholds.

3.1.2 QoE Prediction

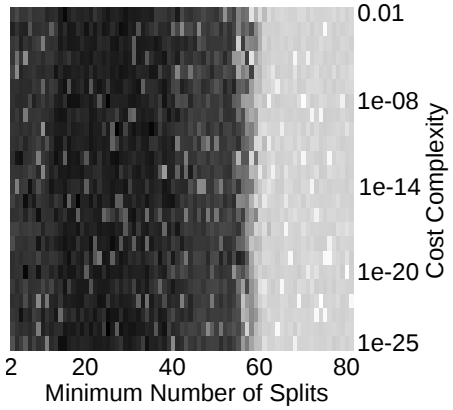
Once the AppQoS indicators are estimated, LEAP can use them to predict a corresponding expected quality of user experience. To establish the relationship between

Figure 3.5: LEAP parameterization: R^2 VS Split



Source: by author (2019).

Figure 3.6: LEAP parameterization: Heat map



Source: by author (2019).

AppQoS and QoE, LEAP employs an adaptation of recent research work, which, based on user interviews, derived a mathematical model for mapping AppQoS to a MOS score. The influence of startup time on MOS is defined by Equation 3.2, where *Qini* is the resulting MOS score and t_0 is the predicted startup time. The values 0.963 and 5.381 were calculated by solving a nonlinear minimization problem for the mean squared error between MOS scores in t_0 and the function$f(t_0)$ (SEUFERT et al., 2014; HOSSFELD et al., 2012).

$$Qini = -0.963 \times \log_{10}(t_0 + 5.381) + 5 \tag{3.2}$$

Additionally, to allow estimation of the effects of stall count and total stall length on QoE, we first need to define a factor λ. This factor represents the ratio between the video's total stall length σ and the total length of the video playback (the sum of σ and the video duration ρ), as shown in Equation 3.3. According to Casas *et al.* (CASAS; SCHATZ; HOSSFELD, 2013), this observation should be done in time slots *T* with a typical duration of one minute. This approach allows generalizing the method for videos

of any length. Once λ has been calculated, it indicates a degradation level $1 \leq i \leq 5$, according to the λ intervals shown in Table 3.1. Finally, MOS *Qst* is calculated according to Equation 3.4, where a_i, b_i, c_i are constants defined $\forall i = 1, 2, 3, 4, 5$ according to Table 3.1, and n is the stall count within a specific time slot T. In Equation 3.4 one can observe that, for a stall count higher than six, the user experience will be fully degraded, resulting in a MOS score 1. We would like to emphasize that the equations presented in this subsection rely on the result of previous research work. The reader interested in their validations may refer to (SEUFERT et al., 2014; HOSSFELD et al., 2012; CASAS; SCHATZ; HOSSFELD, 2013).

$$\lambda = \begin{cases} \frac{\sigma}{\sigma+\rho}, & \text{if } \sigma + \rho < T \\ \frac{\sigma}{T}, & \text{otherwise} \end{cases} \tag{3.3}$$

$$Qst = \begin{cases} 1, & \text{if } n > 6 \\ a_i \times e^{-b_i \times n} + c_i, & \text{if } n \leq 6 \end{cases} \tag{3.4}$$

Table 3.1: QoE prediction: factor values according to λ for MOS Calculation (*Qst*)

Factor	$\lambda < .05$	$.05 \leq \lambda < .1$	$.1 \leq \lambda < .2$	$.2 \leq \lambda < .5$	$\lambda \geq .5$
a	3.01	3.09	3.19	3.24	3.30
b	0.76	0.99	1.52	1.69	1.88
c	1.99	1.90	1.81	1.75	1.69

Source: by author (2019).

3.2 Model Construction

This section presents the practical aspects related to the construction of the prediction model. First, we describe the acquisition of QoS and AppQoS indicators. Right after, we illustrate details of the training environment. Lastly, we examine the resulting prediction models.

3.2.1 Aquisition of QoS and AppQoS Indicators

In order to obtain network performance indicators, we employed an active measurement-based platform named NetMetric (SANTOS et al., 2007). NetMetric works with a "Manager" entity, responsible for configuring schedules to be run in the "Agent" entities. An Agent entity can be used both as the origin of a measurement (source point) or as the target (reflector point). The platform was configured to run groups of two different packet bursts. The first burst makes use of the UDP protocol and unidirectionally measures One Way Delay (OWD), jitter and packet loss by injecting 400 packets of 100 bytes at 50ms intervals. The second burst gauges the throughput using the TCP protocol with 640 packets of 1,488 bytes. Both bursts amount to 992 KB worth of data. Although NetMetric is capable of deriving bidirectional metrics, we have chosen to confine our experiment to unidirectional measurements in the downlink.

In order to construct the ground truth, a specific module was developed for NetMetric in the form of a plugin for the Chrome browser. This plugin enabled the extraction of performance indicators (AppQoS) related to the reproduction of video through an HTML5 native video player. This choice is justified by the migration of big video content providers to HTML5 technology, such as Youtube and Netflix.

In order to measure within the application layer, a NetMetric source Agent entity periodically reproduced videos via the Chrome browser, which retrieved its files from a reflector Agent. Two video files, of one minute each, were used in the algorithm's training phase. Both were in the MPEG v4 format and coded in H.264. While the first had a 720p resolution and 9.2 MB, the second had a 1080p resolution and 14.3 MB. The transfer of videos of different resolutions is deemed necessary in light of introducing different bitrates, which demand varying degrees of performance from the underlying network. Therefore, each of the three AppQoS predictors needed to be trained separately in order to be acquainted with the network demands of each bitrate.

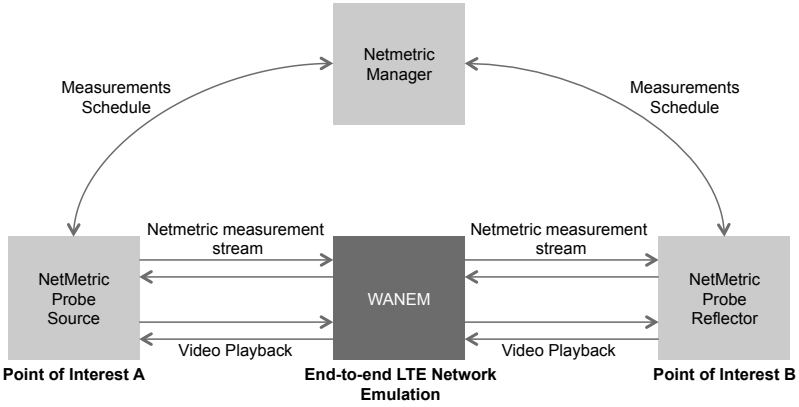
3.2.2 Training Environment

A possible approach for obtaining a dataset relating QoS and AppQoS parameters would be to run the experiment on a deployed LTE network, measuring both the network and application layer indicators simultaneously in an effort to guarantee corresponding conditions in both layers. However, considering the invasive nature of these two

measurement techniques, their concomitant execution would result in mutual interference stemming from a fierce competition for the shared resources. Another possible approach would be to serialize the measurements. Yet, due to a considerable variability in the performance of LTE networks, detectable even in short time intervals, we decided against using such training configuration. The training environment demands stable conditions between the evaluation of the network and application layers. Otherwise, the learning algorithm could establish inaccurate correlations concerning the influence of each network parameters towards the target variables.

In consideration of these difficulties, we opted to devise a controlled environment capable of faithfully emulating the network conditions observed in the deployed setting. In order to reproduce conditions such as throughput, delay, jitter and packet loss present in an LTE environment, we made use of WANEM (KALITAY; NAMBIAR, 2011). The WANEM tool allows us to impose specific constraints on a target network while keeping the network conditions static between consecutive measurements. Figure 3.7 depicts the topology used to set our controlled environment up.

Figure 3.7: LEAP training: controlled environment setup



Source: by author (2019).

The selection of parameters used in the WANEM configuration was based on 7,450 measurements taken by NetMetric in an LTE network deployed countrywide[4], between May and October of 2015. Table 3.2 summarizes the four levels chosen for each of the QoS indicators. The QoS indicators were individually tested as to the normality of their distributions via the Shapiro-Wilk test to a significance level of $\alpha = 0.05$. On account of these normal distributions, the delay, jitter, and throughput indicators have been segmented using quartile analysis. The values associated with the packet loss indicator were selected through the use of modal analysis for integer levels.

[4]For confidentiality reasons, we are not allowed to disclose a detailed characterization of the network.

Table 3.2: LEAP: LTE indicators used in the model training stage

Indicator	Value
Throughput	0.9 Mbps; 8.8 Mbps; 15.5 Mbps; 25.3 Mbps
Delay	22 ms; 56 ms; 64 ms; 98 ms
Loss	0%; 2%; 5%; 13%

Source: by author (2019).

A first round of experiments (via *Full Factorial design* $2^{k.r}$) expressed that the jitter indicator did not contribute to the regression model proposed in the previous section. Once the relevant parameters have been determined (throughput, delay, and packet loss), we performed a subsequent set of experiments, this time with *Full Factorial design* $4^{k.r}$, thus allowing a greater degree of variation for each parameter without incurring in an unmanageably large number of experiments. With three input parameters and four levels, the *design* resulted in 64 possible combinations. After observing the variation in the results, we defined the number of ten repetitions for each combination, considering a significance level of 95%.

3.2.3 Resulting Predictors

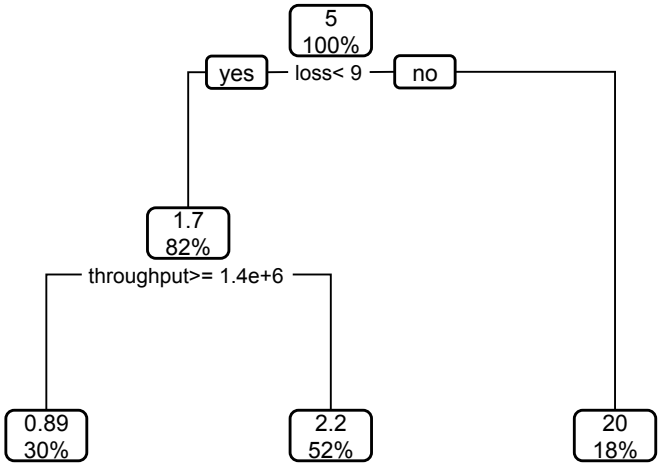
At this stage, after acquiring data from the aforementioned controlled environment, we focused on generating binary decision trees for each of the three AppQoS indicators. These trees were specifically made for the two different resolutions (720p and 1080p). Considering these two details, we had a total of six tree combinations.

The trees represented in Figures 3.2 and 3.3, along with the rest of the trees generated in the scope of this work, display the optimal pruning for these structures. Figure 3.2 shows the tree for total stall length in 1080p resolution as a function of the three primary network indicators. The root node, which represents the first decision, makes use of the throughput as the deciding criterion (represented in bits per second). The value present in the node, that of 2 Mbps, is associated with the bitrate used in the test videos - around 1.9 Mbps. For higher throughput values, the average stall length (0.15 seconds, according to the value indicated by the root left child node) is considerably less than the observed for networks with a lower throughput (75 seconds, as expressed by the node to the right of the root). This observation implies that a network being capable of supplying the video bitrate is the most important factor in determining the overall video behavior.

Another related aspect is the existence of an intermediary range above the bitrate, in which there is a slight degradation for video playback performance indicators - in the case of Figure 3.2, between 2 Mbps and 2.5 Mbps. In this range, instantaneous throughput variations may lead to an empty buffer interruption, which leads to degradation. For values above 2.5 Mbps, however, the video reproduction develops mostly without interruptions, independent of any other predictive indicator. It should be noted that a similar behavior was observed with the 720p resolution, in which the root node was determined by a throughput threshold identical to the video bitrate of 1.2 Mbps, even though a flawless reproduction was only identified for throughput values above 1.9 Mbps.

The root node of the 1080p stall count indicator tree along with the left portion of the tree (not shown in Figure 3.3) are analogous to their aforementioned total stall length counterparts. This means the first decision is taken concerning the throughput in relation to the bitrate, and that higher throughput values presented degradation up to a second threshold (2.3 Mbps). However, the branch shown in Figure 3.3, which represents the portion of the tree immediately connected to the right of the root node, presented a distinct behavior. For this subtree, throughput values between 677 Kbps and 2 Mbps (limited by the root node) predicted an average of 12 stalls, a greater amount than observed for smaller throughput, which averaged 8.4 stalls. An analysis taking total stall length into consideration shows that, for samples with a throughput below 677 kbps, the videos are expected to stay halted for an average of 83.76 seconds. The remainder of cases presented an average of 64.98 seconds. Thus, we can conclude that the video halts for effectively longer in networks with reduced throughput, even though the stall count is smaller, representing longer-lasting individual interruptions.

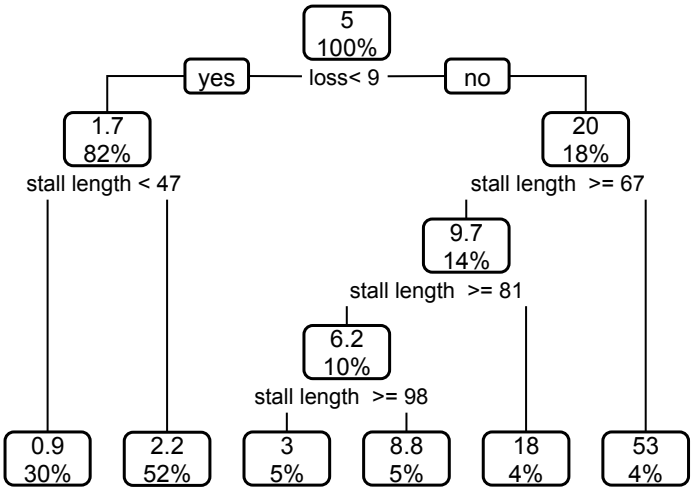
Figure 3.8: Decision tree for startup time of 1080p video playout



Source: by author (2019).

With respect to video startup time, as shown in Figure 3.8, the decision in the root node is taken by a very degraded condition (with packet loss above 9%). According to Table 3.3, these predictor's adjustment factors were low in relation to the measured values, with $R^2 = 0.3147$ in its best case, for 1080p resolution. This hints at an insufficiency of the three network QoS indicators to accurately derive the startup time. In an effort to improve accuracy, we examined which of the involved indicators had the strongest Pearson correlation with the startup time. We observed that, besides the indicators already in use by the decision tree (loss and throughput), the next best-correlated indicator was the total stall length, whose correlation with the startup time has shown to be five times stronger than the one between this indicator and the one-way delay. Figure 3.9 illustrates the decision tree generated considering the total stall length as an additional input variable. The resulting prediction became more accurate to a level of $R^2 = 0.8085$ for 1080p, accompanied by a decrease in root mean square error, as can be seen in Startup Time 2 column in Table 3.3.

Figure 3.9: Decision tree for startup time considering total stall lenght of 1080p video playout



Source: by author (2019).

Table 3.3: LEAP: RMSE and R^2 for 720p and 1080p predictors

Source: by author (2019).

	Total Stall Length		**Stall Count**		**Startup Time**		**Startup Time 2**	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
720p	5.7548 s	0.9627	1.4747	0.8222	4.2321 s	0.2169	2.0151 s	0.8774
1080p	9.8849 s	0.9274	1.1847	0.9449	6.5461 s	0.3147	3.9102 s	0.8085

Source: by author (2019).

3.3 Evaluation

This section presents the performance evaluation of the proposed model. Additionally, this section addresses the results gathered from applying the model to the mobile network mentioned in the Section 3.2.2. Essentially, our objective is to answer the following questions: (*i*) How accurately can the model predict video application behavior based on network performance observations? (*ii*) To which extent is network intrusion kept low when the model is applied to a large-scale LTE network? (*iii*) How can providers capitalize on prediction of QoE?

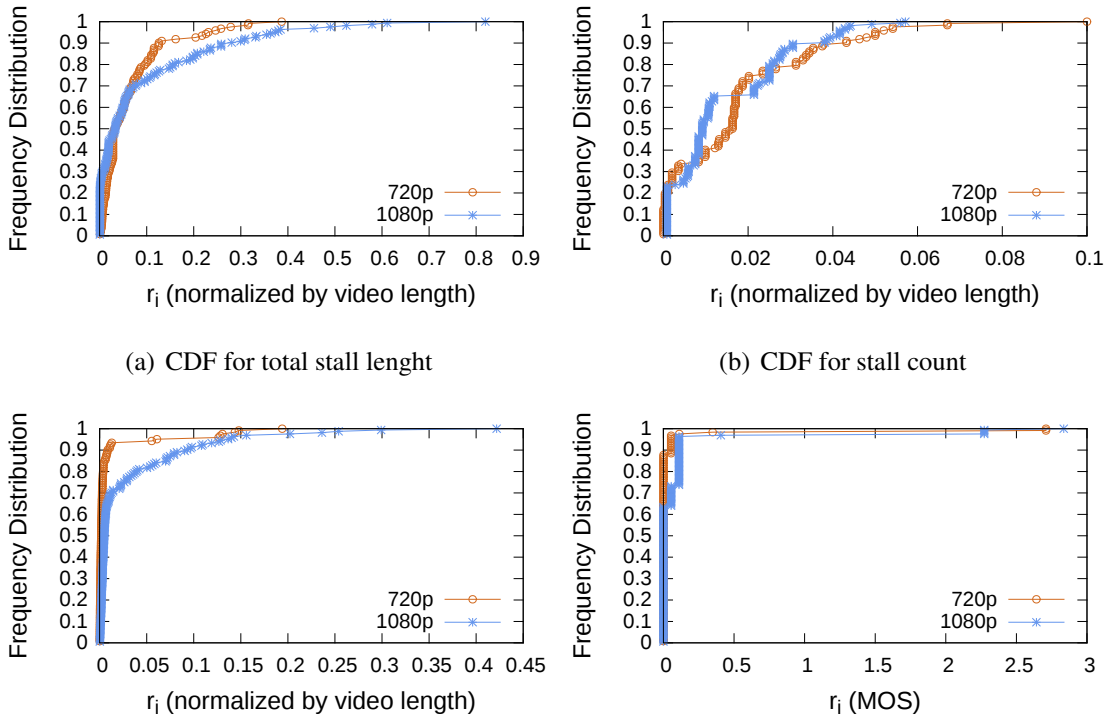
3.3.1 Model Accuracy

With the purpose of determining accuracy, the proposed model was subjected to a test dataset, independent of the training dataset. Each test sample i contains a measurement for each of the three predictor variables (throughput, delay and loss) and three application indicators (stall length, stall count and startup time). By applying the first group of variables as input for the decision trees represented in Figures 3.2 and 3.3, we obtained an estimation for each of the variables in the second group, which then have an associated observed value x_i and a predicted value $\hat{x}_i$. This allowed us to calculate, for each sample i, the normalized residuals r_i, defined by the equation $r_i = |\hat{x}_i - x_i|/N$. We use the factor N to normalize the error values. For each of the three application indicators, the value of N is derived from the duration of the videos employed in the controlled environment (60 seconds in this case), which enables a generalization of the evaluation method for videos of any duration.

Figures 3.10(a), 3.10(b), 3.10(c) and 3.10(d) depict the variation of r_i in the horizontal axis, associating each value in this axis to a portion of the sampling group (in the vertical axis) in which r_i is lesser or equal to the set threshold. Thus, considering the 1080p resolution in Figure 3.10(c) (video startup time), the value of 0.093 in the horizontal axis is related to the 0.9 value in the vertical axis, indicating that 90% of the samples of the corresponding group have $r_i \leq 0.093$. In the same figure, 90% of the samples with 720p resolution have $r_i \leq 0.0086$. This can be interpreted as an error of 9.3% and 0.86%, respectively. The average of r_i in 90% of the samples for all indicators, considering both 1080p and 720p, has a value of $\bar{r_i} = 0.0982$ (9.8%). We deem this error rate to be satisfactory and in line with recent research work.

Based on the predicted and observed values for the application indicators, it is possible to calculate a MOS value according to the approach presented in Subsection 3.1.2. For this indicator (MOS), the following error rates are depicted in an absolute scale, which ranges from 1 to 5. According to Figure 3.10(d), MOS prediction presented an error rate of up to $r_i = 0.11$ for 90% of samples for videos of 1080p. This means that, for example, if a $MOS = 3.5$ is calculated using the observed application indicators, a value of $3.39 \leq \widehat{MOS} \leq 3.61$ is to be predicted 90% of the time. The error rate was smaller in 720p resolution, with a value of $r_i = 0.05$. Even though the AppQoS estimation error rate is low, this disturbance did not significantly impact QoE prediction, which presented an even lower prediction error, as shown in Figure 3.10(d).

Figure 3.10: AppQoS and QoE prediction error for 1080p video playout



(a) CDF for total stall lenght

(b) CDF for stall count

(c) CDF for startup time

(d) CDF for MOS

Source: by author (2019).

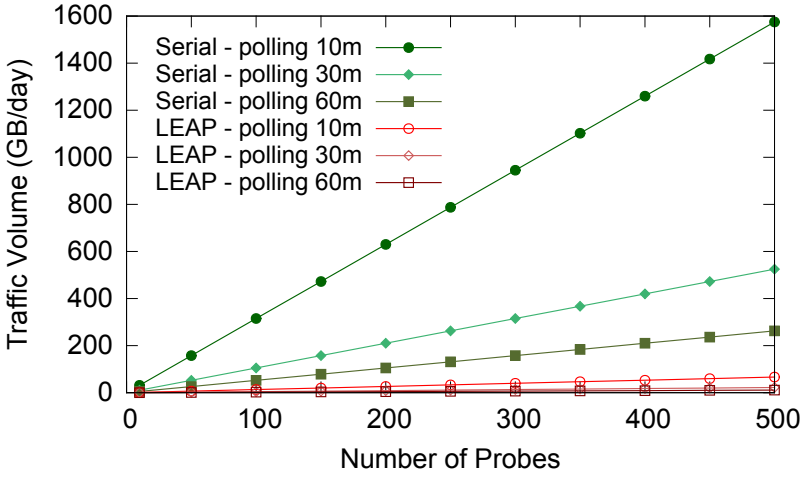
3.3.2 Model Intrusiveness

Figure 3.11 illustrates a comparison of intrusiveness, constructed with traffic volume parameters described in Subsection 3.2.1, between the proposed model and measurements taken through use of real video transfers. It should be noted that the most intrusive

LEAP case (500 probes[5] with a polling interval of 10 minutes) generates 66.65 GB/-day. With an equivalent configuration, the real video transfer strategy (required by full reference quality assessment methods such as SSIM and PSNR) would require 1,574.71 GB/day. Therefore, the adoption of LEAP is on average one order of magnitude less intrusive.

Complementing the aforementioned analysis, take into account that the unstable performance of LTE networks, mainly characterized by user mobility and fierce competition of radio interface resources, demands a reduced monitoring polling interval to achieve a realistic view of the network environment. In this setting, low intrusion is an essential requirement to be met. Otherwise, the generated traffic would consume an enormous amount of resources, undermining the availability of resources for end users.

Figure 3.11: LEAP scalability



Source: by author (2019).

3.3.3 LEAP in Action

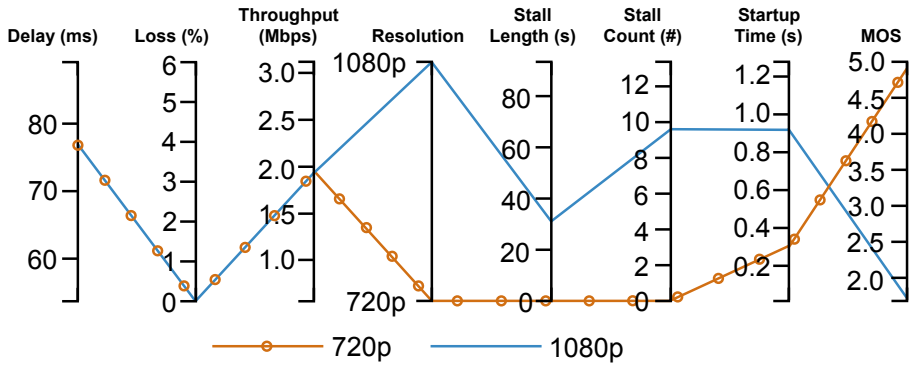
After its creation, the model was deployed in one of the largest mobile operators in Brazil, already mentioned in Subsection 3.2.2. We installed reflector agents as close as possible to the mobile operator's main CDN nodes responsible for delivering cached videos. With this setup, the model was capable of measuring AppQoS and QoE in an end-to-end approach, from an application server down to the end user premises. In order to decide where to install the source agents, we examined the operator's traffic matrix and concluded that a Pareto's Principle fit was reasonable, that is, 80% of all mobile data

[5]The term Probe refers to the packet injected into the network for measurement purposes.

traffic was generated in less than 20% of eNode Bs (a.k.a. Base Station). Then, the source agents were deployed close to each selected eNB in decreasing order of traffic volume.

Here we show a real-world application of LEAP. Using the parallel coordinates visualization technique, Figure 3.12 shows a LEAP graph that illustrates the relationship between QoS, AppQoS and QoE, for a particular source agent. In this figure it is possible to observe that both curves share a QoS condition: 77 ms delay, 0% loss and 1.950 Mbps throughput. However, when LEAP predicts the AppQoS performance and QoE, we verify a very divergent performance between resolutions. For 720p, the predictor estimates zero stall count and a startup time of 290ms, incurring in a MOS of almost 5. However, when predicting 1080p performance for the same network, LEAP predicts a startup time of 910ms, 9.68 stalls with a total length of 32.9 seconds, which implies an MOS score of 1.75. In this particular case (and in other interest points), LEAP provided an integrated, real-time view of what the network is delivering and the corresponding effects on video playout performance, *i.e.*, expected user experience. Moreover, through analysis of the respective decision tree, LEAP was able to provide the notion of how QoS parameters need to be tuned in order to achieve the desired service quality.

Figure 3.12: Integrated QoS, AppQoS and QoE view



Source: by author (2019).

The results obtained suggest that it is feasible to estimate AppQoS and QoE for video streaming applications using QoS indicators as predictors. The estimated parameters achieved an average error below 9.92%, and a MOS estimation error below 11% for over 90% of cases. Furthermore, LEAP requires less than 4% of the traffic volume when compared to traditional techniques. The low intrusiveness allows the service provider to configure systematic measurements without an excessive usage of network resources.

To maintain predictor accuracy over time, LEAP needs to be periodically recalibrated. This event is triggered by a scheduled procedure named Accuracy Check Probe.

This procedure executes a strategically distributed set of reference measurements using the LEAP Browser Plugin and compares the measured AppQoS values against the predicted values. This routine assesses the prediction error and, whenever necessary, triggers the model recalibration procedure.

Now that the QoE prediction model for video streaming is defined, in the next chapter, we introduce a novel heuristic that takes the proposed prediction model as input to provide dynamic QoE-aware path selection.

4 DYNAMIC QOE-AWARE PATH SELECTION

In this chapter, we present a novel scheme for network path provisioning that employs the QoE prediction model LEAP (FILHO et al., 2016) introduced in Chapter 3 to deploy QoE-aware paths in an SDN-enabled[1] mobile network[2]. The proposed approach is able to select and deploy QoE-aware paths in networks characterized by high path diversity, with high accuracy and low network resource consumption. We present a heuristic algorithm capable of finding optimized QoE-aware paths in polynomial time. The proposed algorithm runs on top of an SDN architecture and takes advantage of the northbound interface to coordinate related mechanisms such as QoS active measurements, QoE prediction, and network rule management. We carry out a realistic evaluation by considering real mobile operator topology and video traffic traces, where the proposed heuristic algorithm was found to outperform state-of-the-art approaches.

The remainder of this chapter is organized as follows. In Section 4.1 we present the QoE-aware path selection problem. In Section 4.2 we present SQAPE as well as design aspects regarding path selection in large-scale networks. We conclude this chapter presenting the performance evaluation using realistic topology and workload in Section 4.3.

4.1 Problem Formulation

We start by formally defining the QoE-aware path selection problem. Note that, at this point, we aim to estimate QoE according to a QoS to QoE mapping function. Then, we combine it with flexible SDN-based routing mechanisms in order to maximize the overall MOS. Next, we define the inputs and outputs related to our model.

The physical network infrastructure is represented by a direct graph $G = (N, L)$, where N is the set of nodes (*i.e.*, SDN-enabled forwarding devices) and L is the set of links, such that $L \subseteq (N \times N)$. Links are asymmetric and, therefore, bidirectional links are represented as a pair of arcs ((i, j) and (j, i)). We denote the size of sets N and L

[1] Although this work can be adapted to other underlying technologies, such as BGP and MPLS-TE, the OpenFlow protocol was chosen due to its ability to orchestrate network resources in a centralized manner, with a complete view of the network state, and its abstractions to handle packet forwarding.

[2] This chapter is based on the following publication:

- Roberto Iraja Tavares da Costa Filho, William Lautenschlager, Nicolas Kagami, Marcelo Caggiani Luizelli, Valter Roesler, Luciano Paschoal Gaspary. **Scalable QoE-aware Path Selection in SDN-based Mobile Networks**. IEEE International Conference on Computer Communications (INFOCOM 2018).

by $|N|$ and $|L|$, respectively. Each link $(i, j) \in L$ is associated with QoS measurements (*i.e.*, delay, TCP throughput and packet loss). We assume QoS-related data is gathered from the network infrastructure periodically so as to ensure the accuracy and freshness of our model. Therefore, to each link $(i, j) \in L$ is associated a set of functions defined as follows. Function $D : (N \times N) \rightarrow \mathbb{R}_+$ is used to denote measured delay associated with a given physical link $(i, j) \in L$. In turn, function $T : (N \times N) \rightarrow \mathbb{R}_+$ associates link (i, j) with its maximum measured TCP throughput. Last, function $L : (N \times N) \rightarrow [0, 1]$ associates observed packet loss between nodes i and j.

C defines a set of available video content. An element $c \in C$ represents, for instance, a specific video content. For each content c, a set of offloading locations are known in advance. We represent the set of offloading locations for a given content c as the set P and, therefore, $P(c) \subseteq N$.

A path between two distinct nodes i and j consists of a finite sequence of nodes $\tau = \{n_0, n_1, \cdots, n_h\}$ such that $(n_i, n_{i+1}) \in L(0 \leq i \leq h - 1)$. A path τ is simple if all of its nodes are distinct. We denote a valid path between i and j as $\tau(i, j)$ and its corresponding length by $|\tau|$. The set of all simple paths between i and j is denoted by $\Psi(i, j)$. For ease of presentation, end-to-end QoS measurements of a given path τ are associated similarly to previous definitions. Then, functions $T_p(\tau), D_p(\tau)$, and $L_p(\tau)$ represent, respectively, TCP throughput, delay and packet loss. We derive these QoS indicators similarly to Wang and Crowcroft (WANG; CROWCROFT, 1996). The end-to-end path TCP throughput (*i.e.*, $T_p(\tau)$) consists of the minimum TCP throughput observed over all links in τ. Formally, it is defined as:

$$T_p(\tau) = \mathbf{Min}_{(0 \leq i \leq |\tau|-1)} T\big(\tau[i], \tau[i+1]\big) \tag{4.1}$$

Next, the end-to-end path delay (*i.e.*, $D_p(\tau)$) comprehends the additive sum of measured delays (on links) along the path τ. Therefore, the end-to-end path delay is formalized as:

$$D_p(\tau) = \sum_{(0 \leq i \leq |\tau|-1)} D\big(\tau[i], \tau[i+1]\big) \tag{4.2}$$

Lastly, the end-to-end packet loss estimation (*i.e.*, $L_p(\tau)$) follows a similar strategy to the one proposed by Dobrijevic *et al.* (DOBRIJEVIC; SANTL; MATIJASEVIC, 2015). In their work, the authors estimate end-to-end packet loss based on observed losses in forwarding devices. Here, for higher accuracy, we adopt a strategy that is based on packet

loss observed on links (*i.e.*, including forwarding devices and the physical medium). Similarly to Dobrijevic *et al.*(DOBRIJEVIC; SANTL; MATIJASEVIC, 2015), we multiply observed packet loss along the links used on path τ (Equation 4.3).

$$L_p(\tau) = 1 - \prod_{(0 \leq i \leq |\tau|-1)} 1 - L\big((\tau[i], \tau[i+1]\big) \tag{4.3}$$

Given the above definitions of QoS path composition, we consider our previous work (FILHO et al., 2016) to infer the MOS value for a given path τ. We consider function $\Phi(T_p(\tau), D_p(\tau), L_p(\tau)) \in \mathbb{N}^+$, which correlates end-to-end QoS indicators of a single path τ with the predicted MOS. The Φ two-step function employs decision trees to predict video streaming application performance based on observed network QoS. In a second step, the same function provides a QoE estimation based on the predicted performance for the application layer. The interested reader may refer to (FILHO et al., 2016) for additional information.

Our model considers a set of multimedia connection requests S at a specific time frame t. A multimedia connection request $(i, c) \in S$ represents that a device attached to the infrastructure node $i \in N$ is requesting video content $c \in C$. Notice that content $c \in C$ is potentially available at multiple offloading locations – defined by set $P(c)$. Then, $\forall (i, c) \in S, \forall j \in P(c)$: $\exists (i, j) \in (N \times N)$.

Now we can formally define our problem. Given the inputs of the model, that is, the infrastructure G with its associated metrics and a set S of multimedia connection requests, the problem consists of finding a valid simple path for each $(i, c) \in S$ in given time slot t so as to *maximize* observed MOS (Equation 4.4). Therefore, the output of the model is a set of paths that maximize the overall observed MOS given a set of multimedia requests S.

$$\textbf{Max.} \sum_{\forall (k,l) \in S} \sum_{j \in P(l)} \sum_{\forall \tau \in \Psi(k,j)} \Phi\Big(T_p\big(\tau\big), D_p\big(\tau\big), L_p\big(\tau\big)\Big) \tag{4.4}$$

The model presented in this subsection is used as an important building block of our proposed approach (Section III). SQAPE provides mechanisms to actively monitor the network infrastructure and keeps updated QoS link measurements that are used as input in our model.

4.2 SQAPE

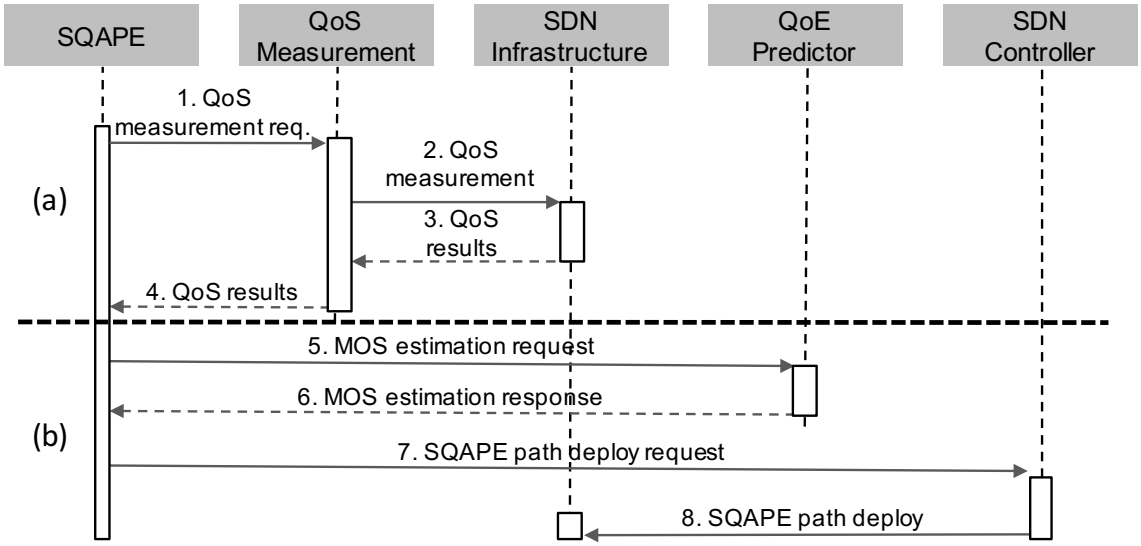
In this section, we present the main aspects of SQAPE. We begin by introducing the SQAPE architecture. Right after, we describe the QoS measurement strategy and the method for QoE estimation. Finally, we present the SQAPE main algorithm and highlight its most essential characteristics.

4.2.1 SQAPE Architecture

SQAPE takes advantage of the centralized control strategy of the SDN/OpenFlow architecture to provide fine-grained control for per-user QoE-aware path deployments in large-scale networks. SQAPE architecture consists of three main components: Path Selection (Decision), QoS Measurement and QoE Predictor (LEAP). These components are decoupled from the SDN controller, which allows integration of any controller instance through the northbound interface (NBI). We claim the proposed scheme to be flexible since it employs loosely coupled microservice-based components and can be orchestrated orthogonally to SDN deployments. For example, both the QoS Measure and the QoE Predictor components are coordinated by the Decision one. Considering the above and in consonance with recent investigations (POULARAKIS et al., 2017; SAHHAF et al., 2017), SQAPE has the potential to cope with incremental SDN deployment strategies.

For didactic reasons, in Figure 4.1 we show the start-up process, which is a special case where SQAPE components interact in a sequential order. After the start-up, events 1-4 run on a periodic basis according to the QoS monitoring frequency (*e.g.*, 60 seconds) and are responsible for determining QoS performance by means of active measurements (Figure 4.1 (a)). In parallel, events 5-8 will occur according to the incoming video requests (Figure 4.1 (b)). At this stage, SQAPE relies on a QoS composition approach followed by a QoS-to-QoE mapping function to compute per-path MOS estimation. This MOS estimation, along with a link utilization heuristic, composes the distance metric which, ultimately, determines the path to be deployed.

Figure 4.1: SQAPE sequence diagram



Source: by author (2019).

4.2.2 From QoS Measurement to QoE Prediction

In this work we consider the use of active measurement methods to assess residual bandwidth, delay and loss. These metrics were selected as the ones most influential to video Quality of Experience according to previous investigations regarding QoS to QoE mapping functions (FILHO et al., 2016; PESSEMIER et al., 2013; HSU; LO, 2014). Active measurement-based methods are particularly interesting as most mobile operators already have tools in place for measuring network QoS indicators.

As formalized in Section 4.1, we perform measurements in the scope of each link and later compose them to form an estimation of the complete path. Using the topology shown in Figure 4.2 as an example, this design choice requires just 94 one-way measurements (*i.e.*, twice the number of links) as opposed to the 22,536 one-way measurements that would be required if we were to measure every possible path between source and destination. As one can observe, the huge number of measurement operations makes path-based approaches impractical for supporting routing decisions in large-scale networks. Also, link-based composition combined with active measurements entails lower network overhead, which is a crucial requirement for scalable monitoring. In practice, per-link measurements require the existence of measurement agents at each vertex. They may be separated by more than a physical link and are represented by the SDN switches, where routing decisions are enforced.

4.2.3 SQAPE Algorithm

The SQAPE algorithm focuses on finding a feasible set of paths with maximum MOS over time. For that, SQAPE attempts to determine a suitable balance between MOS and resource consumption in the infrastructure. One component of SQAPE focuses on MOS maximization, which is a specialization of the *Widest Path problem* (WANG; CROWCROFT, 1996). Another component targets resource consumption, approaching the well-known minimum path problem. Therefore, SQAPE is a hybrid combination of these two algorithms as it considers both metrics simultaneously.

Algorithm 1 consolidates the proposed path selection approach. As one can observe, the measurement phase collects the results of the latest measurement snapshot and updates the QoS graph for each of its links (line 2). The path selection algorithm operates on top of an adaptation of the Dijkstra shortest-path algorithm in which the distance metric is a combination of MOS estimation and a link utilization contribution. The algorithm starts navigating from node k and determines the path with the best local distance metric to each node $u \in N$ (lines 7-14). For each neighbour v, we estimate MOS based on QoS measurement to path τ (line 10-11), according to Function Φ detailed in Section 4.1, and then combine it with an estimation of residual bandwidth (line 12). The objective of this strategy is to capture the interplay among: *(i)* path MOS; *(ii)* path length (hop count); and *(iii)* residual bandwidth. Our algorithm simultaneously maximizes MOS and residual bandwidth while minimizing path length. We achieve this by means of linear combination (line 12). Parameter α determines the importance of the residual bandwidth in relation to MOS. At each iteration, the function Extract-Min removes node u from N according to set χ (line 8). Observe that set χ maintains the best solution found so far, while set φ keeps track of nodes belonging to path $\tau(k, l)$ (line 15). After navigating through all nodes $u \in N$, SQAPE deploys a path for the video request (k, l) such that the $\chi_{[j]}$ is minimum over all $j \in P(l)$. In other words, it takes the minimum path between k and an offloading node j in $P(l)$. Then, we deploy the appropriate SDN forwarding rules (line 16).

Concerning the eventual discrepancy between the frequency of measurements and the frequency of routing requests, the accuracy of path selection can be impaired. The impact of newly routed connections on a path would only be acknowledged after the next measurement iteration, whose frequency may depend on network characteristics. In light of this, after a path is selected, each of its links' residual bandwidth value is

Algorithm 1 SQAPE: QoE-aware path selection algorithm

Input: $G = (N, L)$: network infrastructure
Input: $(k, l) \in S$: video content request
Input: $P(l) \subseteq N$: set of available network nodes (caches) with content l
Input: α: path selection factor; β: throughput estimation factor
Input: M_{max}: maximum MOS value
for every link $(i, j) \in L$ **do**
    gather measured QoS indicator for $T(i, j), D(i, j), L(i, j)$
end for
for every node $u \in N$ **do**
    $\chi_{[u]} \leftarrow \infty$
    $\varphi_{[u]} \leftarrow$ NILL
end for
$\chi_{[k]} \leftarrow 0$
while $N \neq \emptyset$ **do**
    $u \leftarrow$ Extract-Min$\big(N,\chi\big)$
    for each link (u, v) **do**
        $\{T_\tau, D_\tau, L_\tau\} \leftarrow$ QoS metrics of path $\tau(s, v)$ passing through u according to equations (1), (2), and (3)
        $M \leftarrow \Phi\Big(T_p(\tau), D_p(\tau), L_p(\tau)\Big)$
        $\chi' \leftarrow M + \alpha \cdot \Big(\big(\sum_{\forall(i,j)\in\tau_{(k,u)}} \frac{1}{T(i,j)}\big) + \frac{1}{T(u,v)}\Big)$
        $\chi' \leftarrow M_{max} - \chi'$ *iff* $\chi' \leq M_{max}$. Otherwise $\chi' \leftarrow 0$
        if $\chi_{[v]} > \chi'_{[u]}$ **then**
            $\chi_{[v]} \leftarrow \chi'$; $\varphi_{[v]} \leftarrow u$
        end if
    end for
end while
deploy path for req. (k, l) based on $\{\chi, \varphi\}$ such that $\chi_{[j]} : j \in P(l)$ is minimum
QoS update: $\forall(i, j) \in \{\chi, \varphi\}$: *estimate* $T(i, j)$ according to β
return χ, φ

updated to reflect the impact of adding a load comparable to a TCP connection with a given video bitrate (*e.g.*, 1080p, 720p). Considering the video bitrate is determined by the client implementation, our algorithm relies on a parameter β. In this step, the link's residual bandwidth is reduced by a flat amount if there is room for β. If there is no room to subtract β, the residual bandwidth is decreased multiplicatively at a rate proportional to the number of videos recently routed. This method allows connections to be more appropriately distributed when sharing a measurement snapshot.

Note that the problem formulation (Section 4.1) does not explicitly take into account link capacity. However, capacities are considered implicitly as "*soft*" constraints every time MOS values are estimated. It is important to mention that if link capacities (and demands to $(k, l) \in S$) were explicitly considered in the problem formulation (*i.e.*, as "*hard*" constraints), the SQAPE problem would be NP-hard, as it is a generalization of the *Multi-Commodity Flow* problem (LEIGHTON et al., 1995). Last, observe that the proposed algorithm has polynomial time complexity of $O\big(|L| + |N| \cdot log|N|\big)$ using a Fibonacci heap to extract minimum values from χ.

4.3 Evaluation

This section first presents a description of the evaluation environment and its parameters in Subsection 4.3.1. Subsections 4.3.2 and 4.3.3 expand on the first research question: (i) how to accurately predict video QoE for given pairs of source and destinations in large-scale networks? Subsection 4.3.4 addresses the second research question: (ii) how to use the QoE indicator to dynamically select and deploy QoE-aware paths which minimize infrastructure utilization?

4.3.1 Experimental Parameters

The experimental setup[3] consists of: *(i)* a Mininet-instantiated SDN topology, based on a real LTE network deployed countrywide, with four offloading containers mirroring video contents, as shown in Figure 4.2; *(ii)* a realistic video workload comprised of HTTP Adaptive Streaming (HAS) (360p, 720p and 1080p), which is applied to the topology according to each scenario's criteria over 130-minute long experiment rounds; *(iii)* microservice for each of the evaluated algorithms, responsible for selecting and deploying paths on demand; and *(iv)* an active measurement procedure to periodically measure the network state on a per-link basis.

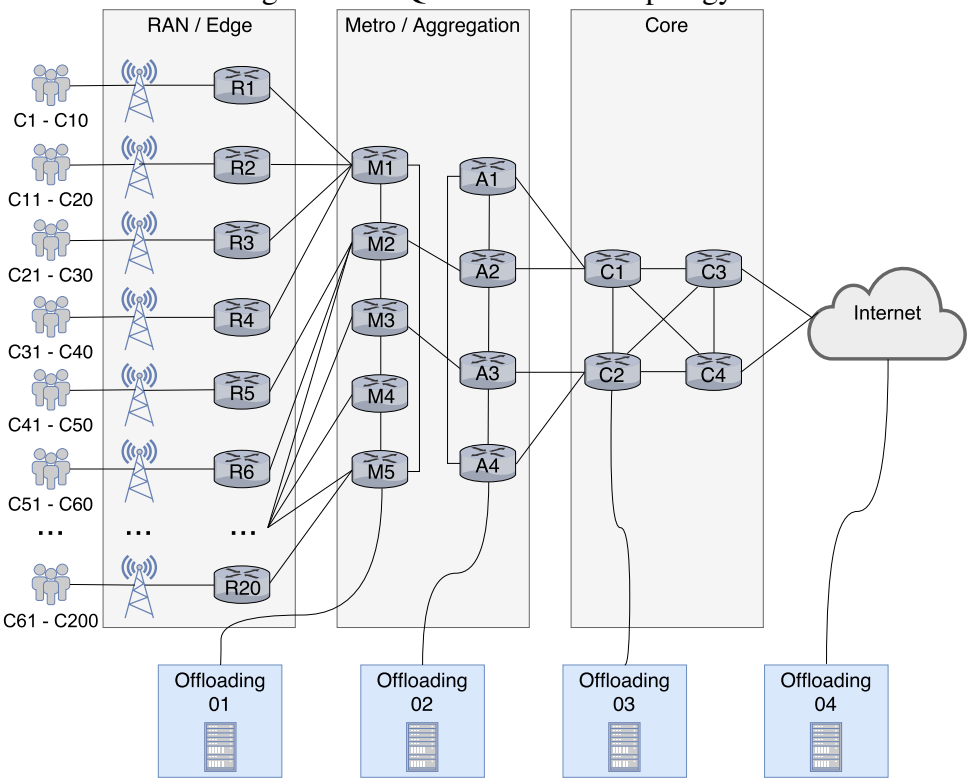
In the performance evaluation, two distinct scenarios were considered. In Scenario 1, SQAPE is compared to a baseline solution, whose path selection is based on administrative weights, usually attributed by a traffic engineer according to a traffic matrix. This kind of static traffic engineering (TE) approach is widely employed in current networks. The weights are adjusted considering an evenly distributed load among the metropolitan layer. The Bellman-Ford algorithm is applied to determine the shortest path by link weight.

In this context, two video loads are evaluated, both approximating the maximum topology capacity. The first load conforms perfectly to the homogeneous premise on which the TE algorithm is based. The second load realistically distributes the videos, including zones with a higher concentration of demand. These different loads aim to contrast how SQAPE and TE adjust to dynamic variations of input.

In Scenario 2, SQAPE is compared to four other path selection algorithms that

[3]The full set of parameters considered in this experiment, including topology and video traces, can be downloaded from https://github.com/rtcostaf/INFOCOM2018

Figure 4.2: SQAPE reference topology



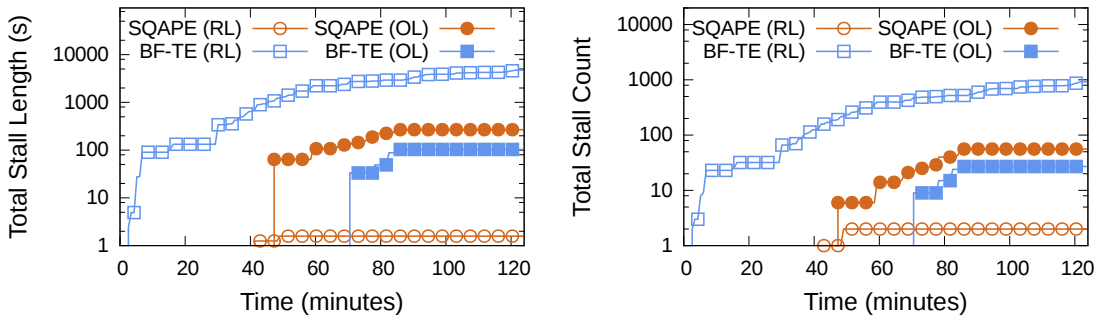
Source: by author (2019).

attempt to minimize one or more network constraints regarding QoS indicators. We considered a workload composed of 744 videos, which were delimited by three components: network capacity, video bitrates and frequency of video requests (following a Poisson fit). The first contender (DKS - Delay) minimizes distance according to the Dijkstra algorithm, using delay as the distance metric (KUIPERS et al., 2002). This technique is commonly employed by routing mechanisms such as OSPF. The second contender (BF - BW) minimizes path's bandwidth bottleneck, using the Bellman-Ford algorithm. This strategy operates iteratively, finding the widest path with the least hops (GUÉRIN; ORDA, 2002; TOMOVIC; RADUSINOVIC; PRASAD, 2015). The third contender is known as the Shortest Widest Path (SWP), which optimally solves the problem of maximizing TCP throughput and minimizing delay in the selected path (WANG; CROWCROFT, 1996). Finally, the fourth contender is the Constrained Shortest Path First (CSPF), which is an adaptation of the Dijkstra algorithm that prunes unfeasible links before performing path calculation (NAM et al., 2014). This algorithm is evaluated differently as it requires an assumption the other proposals do not share: it relies on the possibility of denying video delivery. Thus, it is only graphically presented in Figure 4.6, as other representations depend on the deployment of all videos.

4.3.2 Video Playout

Video stall represents the most important metric to infer the quality of experience of a video playout (NAM; KIM; SCHULZRINNE, 2016; CASAS et al., 2016). In the evaluation procedure, the player was responsible for recording stall count and duration. A stall is detected if the interval $t_i - t_{(i-1)}$ between the conclusion of the download of a video segment and that of the last segment is higher than the remaining buffer (amount of time worth of unconsumed video content available to the user) at time $t_{(i-1)}$. As one can observe in Figure 4.3(a), the static traffic engineering approach, in this case an upper-bound, performed better than SQAPE in its optimal (most-favorable, yet unrealistic) load (BF-TE OL). While TE accumulated 103 seconds of stall throughout 150 videos, SQAPE stalled for 267 seconds. However, in the realistic load (RL), TE behaved considerably worse, accumulating 4,748 seconds while SQAPE stalled for 1.6 seconds. Similar results were obtained for the stall count metric (see Figure 4.3(b)). Thus, SQAPE achieved competitive results considering the load specifically constructed for TE, while SQAPE outperformed TE when subjected to the realistic load.

Figure 4.3: Scenario 1 - Video playout performance evaluation using distinct workloads (Bellman-Ford optimal vs realistic)



(a) Total stall length

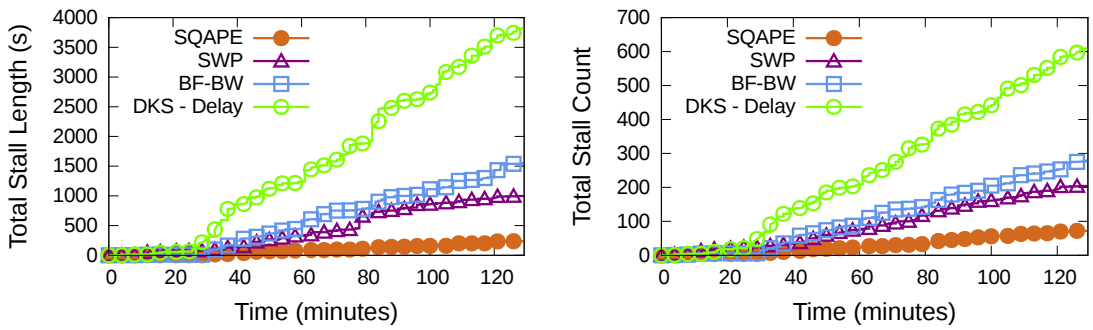
(b) Total stall count

Source: by author (2019).

The potential of SQAPE (w.r.t. adaptability) is more evident in Figures 4.4(a) and 4.4(b), which present the total stall duration and stall count for Scenario 2. In such configuration, SQAPE led to 238 seconds of stall, distributed among 72 stall events. The contender solution with the closest performance (SWP) registered 204 stalls, amounting to 998 seconds of duration. The Dijkstra solution performed the worst, presenting a total of 3,815 seconds during 609 stalls. A few reasons can be given for this result. Firstly, the difference in intrinsic delays between the links causes little initial variation of selected

paths (at this stage, the composition delay is mainly influenced by the propagation delay and the queue delay is minimal), which can lead to an accumulation of several videos in a few links since the early stages of the experiment. This difference can be observed in Figure 4.4(a), where DKS - Delay begins to stall faster than others at around minute 25. Secondly, even though delay is a good predictor for queue occupation, the DKS - Delay algorithm does not consider available bandwidth, which is the predominant indicator for video QoE. The value of considering delay, albeit secondly, is manifested in the difference of outcome between BF-BW and SWP. Although both take the available bandwidth as a priority, SWP takes delay into account, while BF-BW minimizes hop count. This difference can be expressed in BF-BW taking paths with fewer hops but more degraded traffic queues.

Figure 4.4: Scenario 2 - Video playout performance evaluation using realistic workload



(a) Total stall length

(b) Total stall count

Source: by author (2019).

Considering CSPF separately, even though it routed 10.46% fewer videos than the other solutions, SQAPE still managed to accumulate less stall length. It should be noted that denied videos were not accounted into stall and only 7.49% of videos were responsible for all stall events in SQAPE. In light of this, it can be said that SQAPE made better use of the network resources in comparison with CSPF.

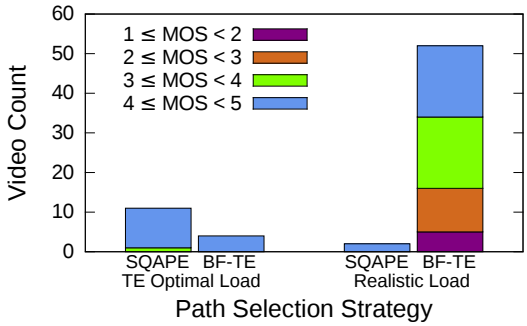
The shortest-widest path algorithm, despite performing well, did not achieve as good a result as SQAPE. SWP usually selects decent paths as available bandwidth and delay are the most influential indicators of QoE, respectively. However, the shortest-widest path is not necessarily the best one. Degradation in indicators other than available bandwidth (such as delay and loss rate) may lower QoE in the widest path. Also, an increase in available bandwidth to levels much higher than the video bitrate does not necessarily lead to an increase in QoE. Further, such an approach does not consider link utilization when determining the path. For example, SWP may prefer to select paths going through

the core, where links are wider, rather than making use of local offloading containers. This tendency can overburden the network core while leaving peripheral links underused. In comparison, SQAPE considers how key QoS indicators influence the final QoE, being aware of paths which may have less available bandwidth but better overall QoE estimation. Finally, since MOS does not increase with available bandwidth indefinitely, SQAPE does not have a preference for paths wider than the expected video bitrate, which allows it to make use of the network edge regularly.

4.3.3 Quality of Experience

The estimated MOS of degraded videos in Scenario 1 is shown in Figure 4.5, considering only impaired videos (*i.e.*, $MOS < 5$). As expected, the traffic engineered solution operated well when applied to its optimal load, resulting in only four degraded videos out of 150, all of which had MOS values considered "good" ($MOS \geq 4$). When applied to the same load, SQAPE degraded ten videos into "good" and one video into "fair" ($3 \leq MOS < 4$). On the other hand, when handling the more realistic load, the traffic engineered solution degraded more than a third of all videos, while SQAPE degraded only two. The TE solution also presented videos in the "bad" range ($1 \leq MOS < 2$), while SQAPE's worst videos were within the "good" range. This result is consistent with the stall values, confirming SQAPE's remarkable adaptability in dynamic environments.

Figure 4.5: Impaired videos by MOS - Scenario 1

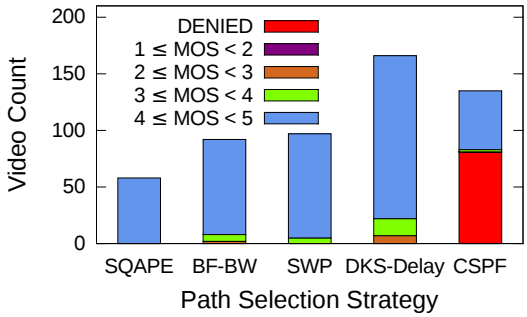


Source: by author (2019).

Regarding Scenario 2, Figure 4.6 illustrates the different amounts and intensities of degraded videos observed for each of the path selection algorithms. The number of videos denied by CSPF is also shown to compare its number of impaired video experiences. In addition to CSPF *denying* more videos than those impaired by our strategy,

it still presented *impaired* videos. The DKS-Delay strategy resulted in the highest observed stall length and most impaired videos, displaying how adopting delay as the sole arbiter of quality fails to determine the best paths. Furthermore, even though BF-BW presented approximately 50% more stall length than SWP, both resulted in a similar number of impaired videos. Instead, this discrepancy was expressed in the degradation intensity experienced by the videos, evidenced by the fact that BF-BW videos reached "poor" ($2 \leq MOS < 3$), differently from SWP. All of these highly impaired videos were observed when retrieved from offloading point 1 (see Figure 4.2), which is the closest content provider. This is consistent with BF-BW's preference for minimizing hop count, which represents a vulnerability depending on the network topology. SWP's auxiliary use of delay was found to be an improvement over hop count.

Figure 4.6: Impaired videos by MOS - Scenario 2

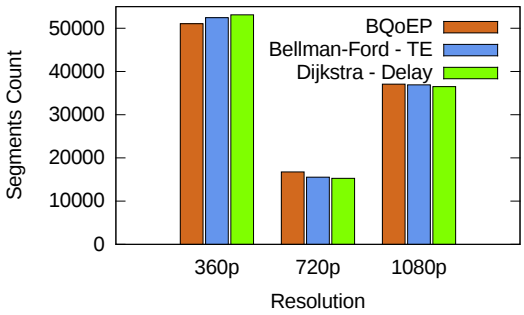


Source: by author (2019).

The performance of SQAPE achieved the lowest video degradation, registering stalls in only 58 of the 774 videos, expressing an improvement of at least 37%. Moreover, none of the stalls were intense enough to lower video MOS below "good" ($MOS \geq 4$). In summary, despite delay being, by itself, insufficient for path selection, it is fundamental for an adequate solution. Furthermore, it is reasonable to conclude that one of the reasons SQAPE outperformed SWP is how it more appropriately incorporates delay as a metric.

Another aspect of Scenario 2 worth discussing is how the HTTP adaptive streaming influences the results. One way to evaluate such effect is to count the number of segments downloaded in each resolution, as shown in Figure 4.7. Even though SQAPE had the edge over other algorithms (with more 1080p and 720p segments), the behavior was comparable throughout the solutions, indicating that no algorithm obtained unfair advantage in traffic demand by favoring lower resolution segments.

Figure 4.7: Segments count by resolution



Source: by author (2019).

4.3.4 Network efficiency

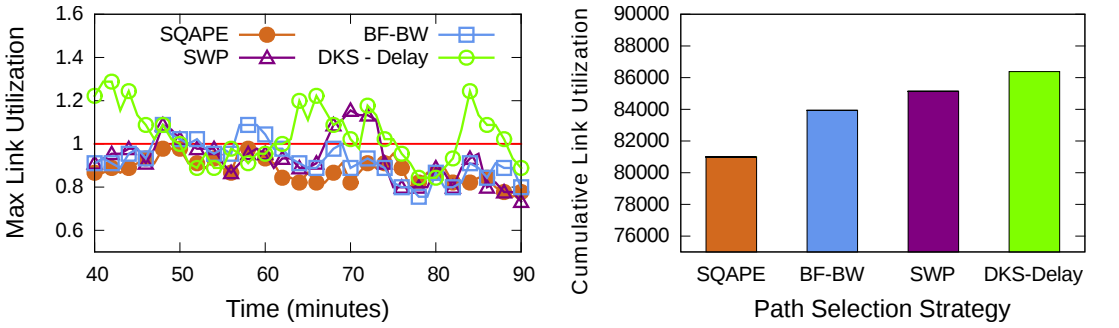
According to a realistic demand, videos of up to 40 minutes of duration are requested continuously over a 90-minute window, resulting in 130 minutes of run time. This produces three different stages. In the initial stage (for the first 40 minutes), the number of ongoing videos increases gradually, while most of the videos have not had the time to finish. The stable stage occurs between minutes 40 and 90, when the video intake is compensated by finishing videos. In the final stage (from minute 90 to 130), no new videos are requested, slowly decreasing network utilization until the last video ends.

In order to determine the efficiency of a solution with respect to network utilization, each link $(i, j) \in L$ is given an indicator $v_{ij}(t)$ that represents, at an instant t, the number of active paths the link takes part. An active path is that which has been designated and deployed for any video running at time t. Considering that the smallest resolution for Scenario 2 is 360p, there is a theoretical limit V_{ij} of how many videos can be supported by the link (i, j), defined by $V_{ij} = c_{ij}/B_{360p}$. Where c_{ij} is the link capacity and B_{360p} is the bitrate of 360p video segments, both in bits per second. When $v_{ij}(t) > V_{ij}$, considering the capacity is divided among the videos being transmitted over a link, it becomes impossible to transfer a new segment in the time it takes to be consumed. This implies the buffer is consumed faster than it grows, generating stall if and when it runs out.

Considering a network snapshot is taken at every minute, let $t \in \mathbb{N}, 0 \leq t \leq 130$. In this snapshot, value $v_{ij}(t)$ is observed for each link $(i, j) \in L$. Occupation rate $R_{ij}(t)$ is given by $R_{ij}(t) = v_{ij}(t)/V_{ij}$. The maximum link utilization at time t, designated as $ML(t)$, is defined as the maximum value of $R_{ij}(t)$ observed in all links, given by $ML(t) = max(R_{ij}(t)), \forall (i, j) \in L$. In Figure 4.8(a), the evolution of $ML(t)$ is shown in the stable stage, for four of the strategies considered in Scenario 2. CSPF cannot be considered in the utilization plot because it deployed a considerably lower number of

videos, not exploring the intricacies of accommodating video streams under adversity. The horizontal line indicates a theoretical limit where $v_{ij}(t) = V_{ij}$ for at least one of the links at the moment t. SQAPE was the only strategy for which the maximum link utilization remained below this threshold throughout the experiment, demonstrating how this solution makes better use of network resources. Using the accumulated v_{ij}, one can estimate path length. By consecutively differentiating between routes deployed by each solution, it is possible to derive an indicator U of utilized links within a snapshot, given by $U = \sum_{t=0}^{130} \sum_{\forall(i,j)\in L} v_{ij}(t)$.

Figure 4.8: Scenario 2 - Network efficiency using realistic workload



(a) Maximum link utilization histogram (b) Cumulative link utilization based on path length

Source: by author (2019).

The value of U for each algorithm gives insight into how long are the selected paths during the experiment. Keep in mind that the longer the paths, the fewer resources are available (which could be used to transmit other contents otherwise). As shown in Figure 4.8(b), SQAPE had the best results in relation to the other algorithms. In a joint analysis of the plot in Figure 4.8(a), it can be stated that SQAPE did not require longer paths to avoid network bottlenecks. Also, solutions that made use of delay as a criterion frequently opted for longer paths. This phenomenon can be explained by congested links presenting higher delay and paths taking a detour to avoid them, resulting in more link usage. Such paths may offer less degradation. However, higher link utilization tends to deplete the network resources more quickly, as can be seen in Figure 4.8(a), where both SWP e DKS - Delay are also the ones with the worst bottlenecks $ML(t)$.

In this chapter, we combined the QoE prediction model with a polynomial time heuristic for providing QoE-aware path selection. This approach partially answers Research Questions 1 and 2, because it deals only with traditional video streaming. Next, we address the context of the VR videos by introducing a QoE prediction model and an experimentation platform in Chapters 5 and 6, respectively.

5 PREDICTING THE PERFORMANCE OF VR VIDEO STREAMING

In this chapter, we present PERCEIVE (PERformanCe EstImation for VR vidEos), a two-stage adaptive VR performance assessment model. Inspired by the model introduced in Chapter 3, it employs machine learning algorithms to first predict VR video playout performance, using network QoS indicators as predictors[1] (FILHO et al., 2018). Then, along with a novel QoE model for the VR context, it uses the video playout performance metrics to model and estimate the end-user perceived quality. Evaluated in real-world 4G/LTE network conditions, PERCEIVE not only accurately predicts the VR videos performance, but also allows us to pinpoint the QoS conditions that affect VR streaming services the most.

The remainder of this chapter is organized as follows. In Section 5.1, we introduce the approach used for the tile-based adaptive VR video streaming. In Section 5.2, we describe PERCEIVE, the two-step performance prediction scheme that we propose. Finally, in Section 5.3, we report the evaluation carried out to prove concept and technical feasibility of the proposed approach. It includes details on the evaluation methodology, the generation of the dataset, analysis of the training set and results.

5.1 Adaptive Streaming of VR videos using Tiles and Quality Zones

To cope with 2D adaptive streaming techniques, VR videos must be encoded at different quality levels, temporally divided in segments and spatially tiled (NIAMUT et al., 2016). Then, during the streaming session, only the tiles within the viewport are streamed in high quality, while others are maintained at the lowest levels or not streamed at all (QIAN et al., 2016). To be effective, these techniques rely on viewport prediction algorithms, since the player needs to fill in a playout buffer with tiles that are expected to compose the viewport in the near future (PETRANGELI et al., 2017).

Although the use of viewport-aware techniques leads to the reduction of bandwidth consumption, the effects of network performance on VR video streaming still plays

[1] This chapter is based on the following publication:

- Roberto Iraja Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck, Luciano Paschoal Gaspary. **Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks**. ACM Multimedia Systems Conference (MMSys 2018).

an important role on the user's perception of the services. Since the full panoramic view of a VR video usually demands a much higher bitrate, when compared to regular videos (CORBILLON et al., 2017), even a fraction of it (viewport) may require high bitrates. Along these lines, recent investigations have emphasized the importance of the network effects on the perceived quality (Quality of Experience, QoE) of adaptive video streaming applications (MAO; NETRAVALI; ALIZADEH, 2017; JIANG et al., 2016; JIN et al., 2016; DIMOPOULOS et al., 2016; FILHO et al., 2016). However, state-of-the-art approaches fall short in predicting the perceived quality for VR videos as they do not consider the spatial segmentation (*e.g.*, tile-based videos).

QoE has shown to be a critical factor for video applications (NAM; KIM; SCHULZRINNE, 2016; AHMAD; FLORIS; ATZORI, 2016). As such, both network operators and VR content providers are required to answer an important question: *considering the wide range of performance levels of IP networks, to which extent are the currently observable network conditions able to provide users of VR applications with adequate QoE?* Answering this question is remarkably complex due to two constrains. First, the influence of the network on VR video performance is unknown; and second, the state-of-the-art on video QoE estimation modeling does not consider the VR context.

In order to reduce the bandwidth required for the streaming, PERCEIVE adopts a tiling structure, in which the videos are not only divided in temporal segments but are also spatially split in sections (tiles) (PETRANGELI et al., 2017). In addition, tiles are grouped in quality zones prior to the streaming. Each of the zones is assigned a quality level according to the network conditions measured during the previous segment. In the next two subsections, both the structure and the adaptive streaming technique adopted in this work are presented.

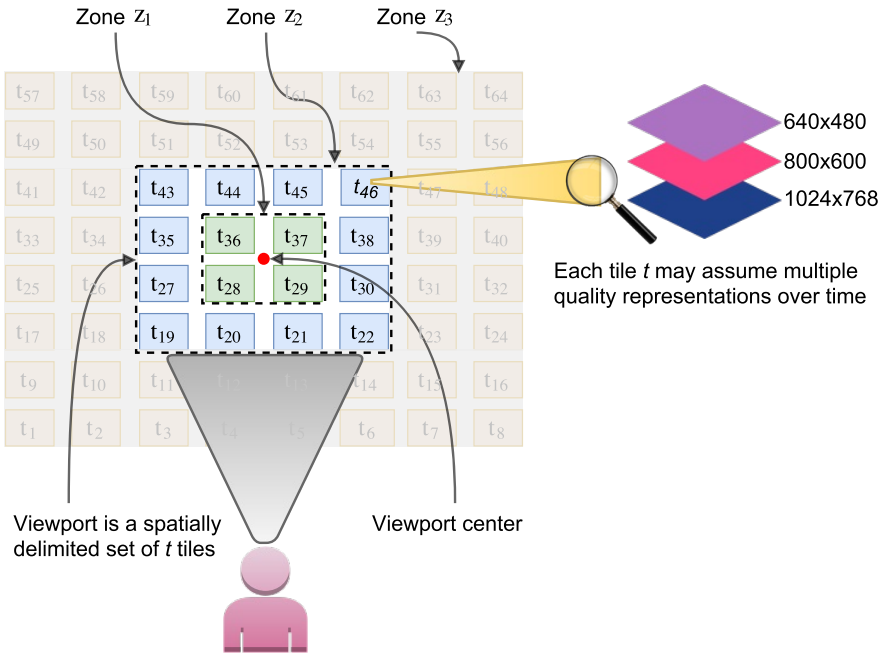
5.1.1 Adaptive VR streaming structure: Spatial Tiles and Quality Zones

A VR video V can be represented by a set of k spatially divided zones $Z = \{Z_1, ..., Z_k\}$ such that $\bigcap_{\forall k} Z_k = \emptyset$. The same video V is temporally split into a discrete number of m segments $S = \{S_1, ..., S_m\}$ such that $\bigcup_{\forall m} S_m = V$. Each zone Z_k is composed of a set of tiles $t \in Z_k$. A tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$, and may assume different bitrates (qualities) $R(C_{t_m})$ over time. Finally, we refer to a segment as the set of all chunks for a given time frame such as $S_m = \bigcup_{\forall t} C_{t_m}$. In tile-based approaches, the encoding process defines how the video will be spatially divided

(*i.e.*, tiling scheme), which bitrates will be available in the HAS context (*i.e.*, quality representations), and the segment length (*i.e.*, number of seconds).

An example of this type of structure is shown in Figure 5.1. There are three quality zones $Z = \{Z_1, Z_2, Z_3\}$, each one composed by a set of adjacent tiles. Z_1 is a set of tiles adjacent to the viewport center $(t_{28}, t_{29}, t_{36}, t_{37})$, Z_2 is the border of the viewport $(t_{43}, t_{44}, t_{45}, t_{46}, t_{38}, t_{30}, t_{22}, t_{21}, t_{20}, t_{19}, t_{27}, t_{35})$ and Z_3 is composed by all tiles outside the viewport.

Figure 5.1: Example of an adaptive tile-based VR video structure split in 3 quality zones.



Source: by author (2019).

5.1.2 Adaptive streaming heuristic

Algorithm 2 describes the adaptive streaming heuristic procedure adopted for this work (PETRANGELI et al., 2017). The bitrate in a specific zone Z_k is named as $R(C_t)|^{Z_k}$. The algorithm receives as input *(i)* a reference to a VR video V, *(ii)* a set S describing the video segmentation and *(iii)* the available zones in video V. The heuristic described in Algorithm 1 works as follows. Once knowing the available bandwidth in the network, the VR player downloads tiles with the highest possible bitrates. First, the heuristic tries to increase the bitrates on the zones inside the viewport. Then, it repeats the procedure to tiles from the outer zones (observed in line 3). Observe that the heuristic does not increase

the bitrate $R(C_t)|^{Z_{k+1}}$ on a subsequent zone Z_{k+1} in case the bitrate of zone Z_k is strictly upper-bounded by $R(C_t)|^{Z_{k+1}}$. In other words, it ensures that the bitrate of zone Z_{k+1} is always lower or equal to that of zone Z_k. Furthermore, it ensures that tiles within the same zone Z_k are streamed with the same bitrate $R(C_t)$, that is, $R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$. If the available bandwidth were insufficient to download all the tiles in a zone on time (before display), the streaming would stall until the buffers were filled. Hence, the player ensures that all tiles are synchronized during the playout and that no black tiles appear. For more information on the working principles of the streaming heuristic, please refer to (PETRANGELI et al., 2017).

Algorithm 2 PERCEIVE: VR player heuristic (PETRANGELI et al., 2017).

Input: V: VR video
Input: S: discrete number of segments in VR video V
Input: $Z = \{Z_1, ..., Z_k\}$: k spatially divided zones in VR video V

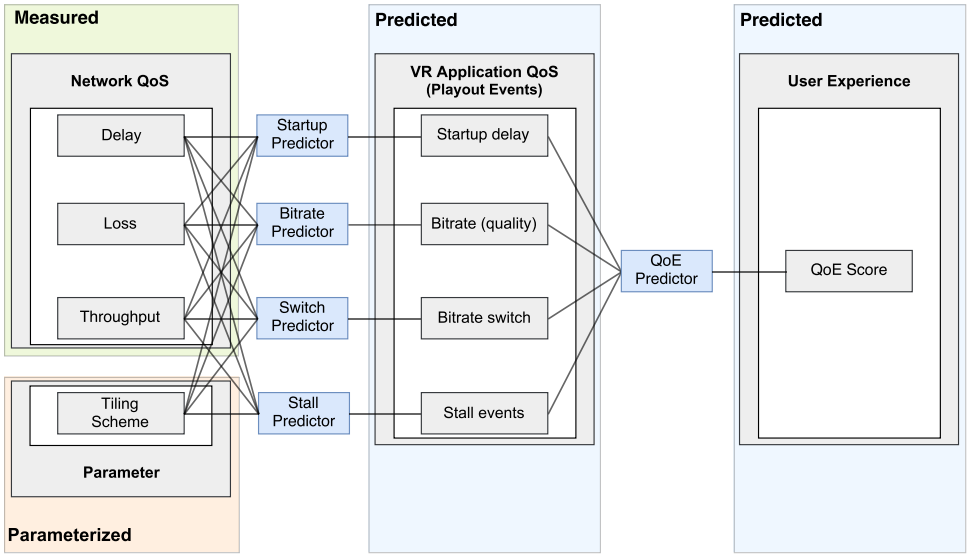
1: **for** each video segment $S_i \in S$ from VR video V **do**
2: **for** each zone $Z_k \in Z$ **do**
3: *gather* tiles $t \in Z_k$ with maximum available bitrate $R(C_t)|^{Z_k}$, such that $(\forall k \geq 2) : R(C_t)|^{Z_k} \leq R(C_t)|^{Z_{k-1}}$ and $(\forall t \in Z_k) : R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$
4: **end for**
5: **end for**

5.2 PERCEIVE: Adaptive VR video performance prediction

Figure 5.2 presents the block diagram of the proposed two-stage VR performance prediction method. The first stage is composed of four predictors, one per VR video application performance metric (*i.e.*, startup delay, quality, quality switches count and video stalls) (YIN et al., 2015b). As input, the predictors consider both the network Quality of Service (QoS) (*i.e.*, delay, packet loss and TCP throughput) and the tiling scheme. In the second stage, the user QoE is estimated by submitting the predicted application layer performance metrics to the proposed QoE model. PERCEIVE can dissect VR video playout performance by understanding two key processes, namely *(i)* the influence of the network performance on VR video player outputs; and *(ii)* how the user perceives the resulting video playout performance. However, both the VR video player and the QoE model are open questions in the sense that there is neither a reference player implementation nor a QoE model for VR videos. Considering the above, the proposed two-stage

prediction allows both the playout performance metrics predictors and the QoE model to be individually updated, without the need to rebuild the entire scheme. The following two subsections provide details and insights on each of the stages of which PERCEIVE is composed.

Figure 5.2: PERCEIVE two-stage quality prediction



Source: by author (2019).

5.2.1 Adaptive VR Video Playout Prediction

In the first stage of the method, the four most important playout performance metrics associated with the adaptive VR streaming, namely startup delay, quality level (bitrate), quality switches and stall time (YIN et al., 2015b), are predicted based on network QoS inputs and the VR video structure. Each of them is independently predicted using regression trees as predictors (taking advantage of and adapting the 2D procedure proposed in previous research work (FILHO et al., 2016)). Regression trees are employed due to three main reasons. First, they have shown to be an accurate machine learning method in related investigations (SHAFIQ et al., 2014; FILHO et al., 2016). Second, they permit understanding complex and non-linear relationships between predictors and response variables in an intuitive and visual manner. This is a very important feature that allows to pinpoint the most influential inputs, which will be put to the test in the analysis of Section 5.3.3. Finally, once the prediction structures are generated, they can predict the response variable in linear time complexity, and can be easily integrated in third-party

applications, which are fundamental aspects for network operators and content providers.

The selection of the input QoS parameters has been made based on state-of-the-art research studies on the QoS conditions that affect video streaming services the most (VEGA; PERRA; LIOTTA, 2018; PAUDYAL; BATTISTI; CARLI, ; FILHO et al., 2016). These studies also concluded that TCP throughput is one of the most influential QoS metric when predicting QoE. Also, both network losses and delays have been demonstrated to be responsible for further degradation, depending on the type of streaming application used. In addition to these three network performance metrics, a fourth parameter, namely the tiling structure of the VR streaming, was included. The structure defines the number of tiles that need to be streamed to the client, thus it will heavily influence the VR playout performance. Once the four VR playout performance indicators are predicted, they serve as input to the second phase, the QoE model as it is presented in the next Section.

5.2.2 Adaptive VR QoE Estimation Model

The purpose of this second stage is to estimate the quality perceived by the end users (their QoE), considering the VR video playout performance metrics obtained from the previous stage of PERCEIVE (*i.e.,* startup delay, quality level (bitrate), quality switches and stall time). The model proposed herein considers the state-of-the-art on QoE modeling for adaptive streaming applications in general and HAS in particular (PETRANGELI et al., 2015; YIN et al., 2015b; MAO; NETRAVALI; ALIZADEH, 2017). To the best of our knowledge, this is the first model to consider the concept of zones and tiles in a QoE estimation model for VR videos. These characteristics are crucial, given the fact that they allow coping with VR video attributes while providing flexibility to handle different video encoding strategies (*e.g.*, tiling scheme, viewport geometry and available quality representations).

Given the concepts of quality zones and tiles of the approach used for Adaptive VR streaming (Section 5.1), the QoE function is defined per zone as a function of the VR playout characteristics predicted by the previous stage within that quality zone (Section 5.2.1). This strategy is aligned with the notion that the influence of VR playout characteristics on user perception is different depending on the zone where they are observed (*e.g.*, quality switches for tiles outside the viewport are less important than quality switches inside the viewport). Thus, the per-zone quality function ($\phi(Z_k)$) is defined as

the weighted sum of the four playout characteristics (Equation 5.1).

$$\phi(Z_k) = \overbrace{\sum_{\forall t \in Z_k} \sum_{\forall c \in C(t)} q(R(C_{t_m}))}^{\text{Quality}} - \overbrace{\mu \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+}^{\text{Stalls}} - \underbrace{\lambda \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left| q(R(C_{t_m+1})) - q(R(C_{t_m}) \right|}_{\text{Quality switches}} - \underbrace{\omega \cdot T_s}_{\text{Startup}} \tag{5.1}$$

In Equation 5.1, $R(C_{t_m})$ R represents the bitrate (*i.e.,* quality) of a given chunk. Recall that a tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$ (YIN et al., 2015b; MAO; NETRAVALI; ALIZADEH, 2017). Function q is a mapping function that translates the bitrate of chunk C_{t_m} belonging to tile t to the quality perceived by the user (*i.e.*, in terms of bitrate sensitivity). The second term of the Equation is used to track stall time. Stalls can be characterized either by tile (*i.e.*, it is possible to have stall in some tiles and video playout in other, for the same segment) or by segment (*i.e.*, the video will stall until all the tiles for a given segment have enough buffer). To keep the model as general as possible, we consider for each chunk c, that a stall event occurs when the download time $\frac{d_c(R_c)}{C_c}$ is higher than the playout buffer length (B_c) when the chunk download started. Hence, the total stall time is given by $\sum_{c=1}^{C} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+$. In addition, $|q(R_{ct+1}) - q(R_{ct})|$ considers the quality switches between consecutive chunks and T_s tracks the startup delay. Finally, constants μ, λ, ω are the non-negative weights used to tune the model for different user importance regarding QoE events. For example, a higher value of μ, with respect to the other weights, means that the user is more susceptive to video stalls. Consequently, these events should affect the QoE indicator more severely.

Each of the zones within the VR video influences the perception of the user in a different manner. For example, tiles within the first or second zones (*i.e.*, the closest to the FoV of the user) will greatly steer the quality perceived by the user, while bad qualities on tiles of the edge zones will potentially go unnoticed. For this reason, the overall video QoE ($\phi(V)$) is modelled as a weighted linear sum of the QoE measurement per zone (Equation 5.2). Each weight ($\alpha_1, \alpha_2, ..., \alpha_k$) allows defining the relative importance of each zone when composing the video QoE. For example, the zones belonging to the viewport should have higher weights compared to the other zones.

The values for each α_n parameter should be derived from subjective tests. For

example, considering a two-zone QoE scheme, values for α_1 (viewport) should be close to one, and values for α_2 (outside viewport) should be close to zero. When the QoE model is configured with more than two zones, it is necessary to determine α_n (testing values within a certain range) for each zone. In this case, subjective tests should systematically include incremental quality degradation, specifically in the intermediate zones, in order to measure the user's sensitiveness regarding quality issues in each zone.

$$\phi(V) = \alpha_1 \cdot \phi(Z_1) + \alpha_2 \cdot \phi(Z_2) + ... + \alpha_k \cdot \phi(Z_k) \tag{5.2}$$

5.3 Evaluation

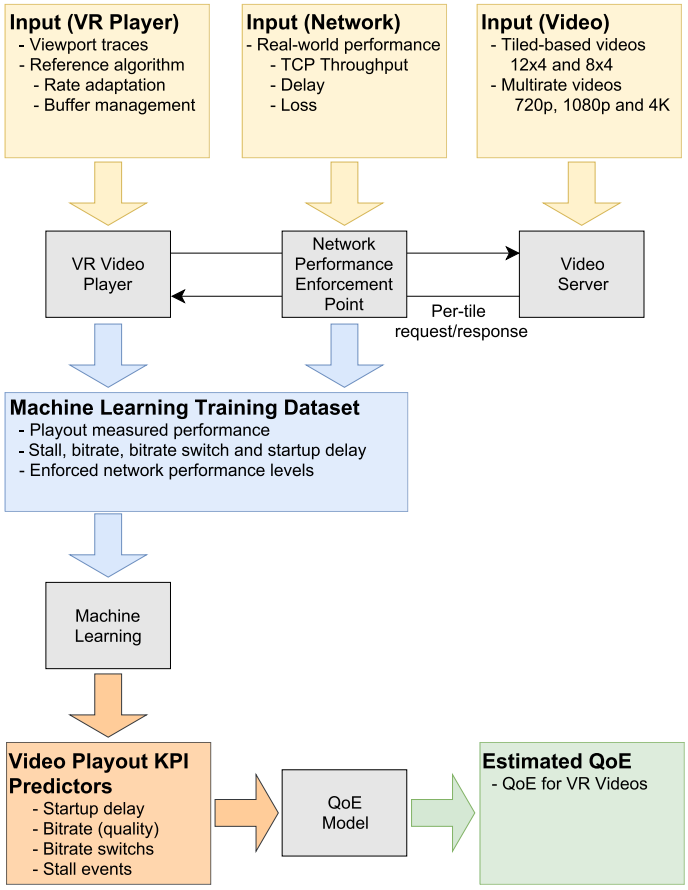
In this section, the evaluation of the PERCEIVE method is discussed. We start by presenting the procedure followed to evaluate the method in Section 5.3.1. Next, in Section 5.3.2, we introduce the generation of the dataset used for training and testing. In Section 5.3.3 we discuss and analyze the resulting VR playout predictors. This analysis provides insights on the dependency and predictability of each of the VR playout performance metrics given the QoS and tiling structure inputs. Finally, in Section 5.3.4 we present the prediction evaluation results for each of the five outputs of PERCEIVE (*i.e.*, the four VR playout performance metrics and the perceived quality).

5.3.1 Evaluation Methodology

In order to evaluate the performance of PERCEIVE, the procedure outlined by Figure 5.3 is followed. First, the datasets for training and testing have to be generated. Therefore, a VR video player is required to measure the VR video application playout performance metrics (*i.e.*, startup delay, average bitrate (quality), bitrate switches and video stalls) while subjected to real-world inputs, such as a realistic wireless networks measurements, VR tile-based videos and users' head track traces.

Next, the resultant datasets are given as input to the machine learning algorithm (responsible for learning the influence of the network QoS parameters and tiling scheme onto the VR playout characteristics). After the training phase, the resulting predictors

Figure 5.3: General evaluation methodology for PERCEIVE



Source: by author (2019).

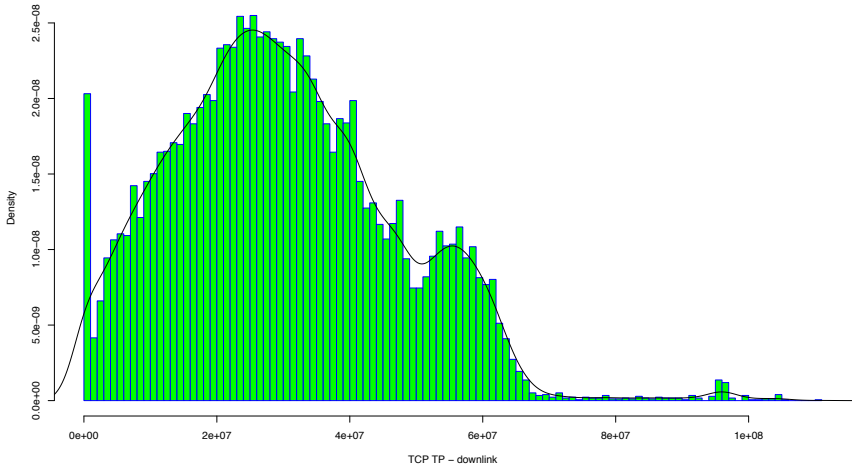
can estimate the application layer performance only by means of the network parameters, and the considered tiling scheme. Finally, based on the VR playout performance metrics, the QoE can be estimated. The performance of PERCEIVE is assessed by means of the calculation of the normalized residual errors between predicted and measured values (r_i, Equation 5.3). In the equation, x is the ground truth, $\hat{x}$ is the prediction and N is the normalization factor (in this case the video duration).

$$r_i = |\hat{x}_i - x_i|/N \tag{5.3}$$

5.3.2 VR Dataset Generation

Each sample in the dataset contains the VR video tiling information, the three network QoS features and the respective video performance measured by the VR video player. To construct such dataset, the procedure presented in Section 5.3.1 is followed.

Figure 5.4: TCP throughput histogram of the 4G/LTE dataset of (HOOFT et al., 2016)



Source: by author (2019).

Experiments are set, considering that a VR video player requests and processes tile-based VR videos from a web server (Apache 2 2.4.18-2). The network conditions are enforced by the Linux Traffic Control (TC) mechanism according to real-world network performance inputs. The experiments are built on top of a Linux Ubuntu 14.04 operating system, running on bare metal servers, where each server consists of a quad-core E3-1220v3 (3.1GHz) processor, with 16GB of RAM and two 10-gigabit network interfaces. Considering this infrastructure setup, we performed 1,524 video execution rounds, which resulted in more than 5,240 minutes of VR video playout.

Table 5.1 summarizes the input parameters values considered in the experiments. As network throughput input, the 4G/LTE measurements dataset of van der Hooft *et al.* (HOOFT et al., 2016) was selected. This dataset presents TCP throughput ranging from 0 Kbps to 95 Mbps as shown in Figure 5.4. For network packet loss, values between 0% and 13% were selected, in line with (FILHO et al., 2016). The network delay range was set from 1 to 130 ms. These values allowed us to assess the application performance from a very degraded delay performance (130 ms) down to the expected 5G delay (1 ms) (DAHLMAN et al., 2014).

Table 5.1: PERCEIVE dataset: input parameter configurations.

Metric	Short	Unit	Range
TCP throughput	TCPTP	Mbps	0-95Mbps ((HOOFT et al., 2016), Figure 5.4)
Packet Loss	PLR	%	$0 - 13\%$ (based on (FILHO et al., 2016))
Delay	Delay	millisecond	1-130ms (based on (FILHO et al., 2016; DAHLMAN et al., 2014))
Tiling scheme	Tile	categorical	8×4 or 12×4 (based on (KOHAVI et al., 1995; QIAN et al., 2016)

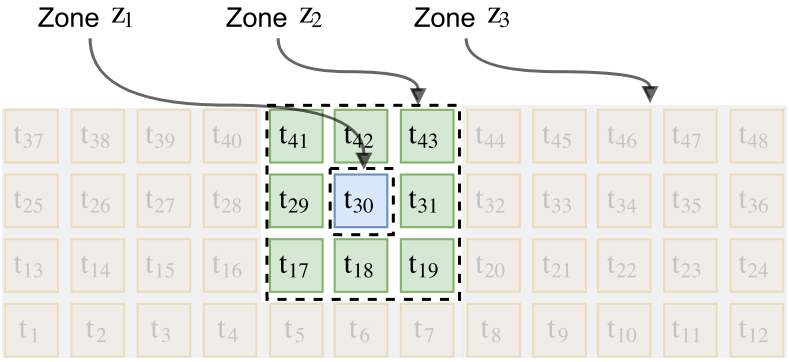
Source: by author (2019).

Two VR videos from Wu et al.'s dataset (WU et al., 2017) (namely "Google

Spotlight-HELP" and “Freestyle Skiing") were used for the streaming under the above described network conditions. For each video, we considered the available datasets regarding the users' head movements while watching it. As the original videos are not tile-based, they had to be re-encoded. After extracting the raw YUV files, making use of the Kvazaar encoder (VIITANEN et al., 2015), the videos were re-encoded in a HEVC/H.265 tile-based version, considering two tiling schemes: 8×4 and 12×4 (QIAN et al., 2016; KOHAVI et al., 1995). In addition, each tiling scheme was encoded to three quality representations, namely 720p (1.8Mbps), 1080p (2.7Mbps) and 4K (6Mbps).

Next, we used the MP4Box[2] application to pack the re-encoded videos into MP4 containers. Finally, we defined the segment duration of 1 second and used MP4Box to extract per-tile files and to generate the MPEG Dash Media Presentation Description (MPD) files considering multiple quality representations (Table 5.2). For the streaming heuristic (Section 5.1.2), there are three defined zones, where Zone 1 is the viewport center tile, Zone 2 groups the 8 tiles surrounding Zone 1, and all other tiles belong to Zone 3. Figure 5.5 shows the zone division for the 12×4 tiling scheme.

Figure 5.5: Viewport detail for the 12x4 tiling scheme



Source: by author (2019).

Table 5.2: PERCEIVE dataset: adaptive streaming configurations.

Videos	Qualities (bitrates)	Quality zones	Segment	Tiling
Google Spotlight Freestyle Skiing (Wu et al. (WU et al., 2017))	720p - 1.8Mbps 1080p - 2.7Mbps 4K - 6Mbps	Zone 1: 1 tile (central FoV) Zone 2: 8 tiles (adj. Zone 1) Zone 3: Rest	1 s	12×8 8×4

Source: by author (2019).

[2]MP4Box https://gpac.wp.imt.fr/mp4box/

5.3.3 Resulting Predictors: VR Playout vs Network Conditions

Based on the dataset, the regression trees were trained using a 10-fold cross-validation approach (KOHAVI et al., 1995). As each zone has independent quality behavior (bitrate), both the quality and quality switches need to be learned per-zone. On the other hand, the startup and stall times are independent from the quality zone under scrutiny. Hence, they can be learned per video segment. Given the fact that there are three quality zones, eight regression trees were trained: three for Quality, three for Quality Switches, one for Stall time and one for Start time. All trees are optimally pruned (MURTHY, 1998), which means pruning until the cross-validation error is minimal and overfitting is avoided.

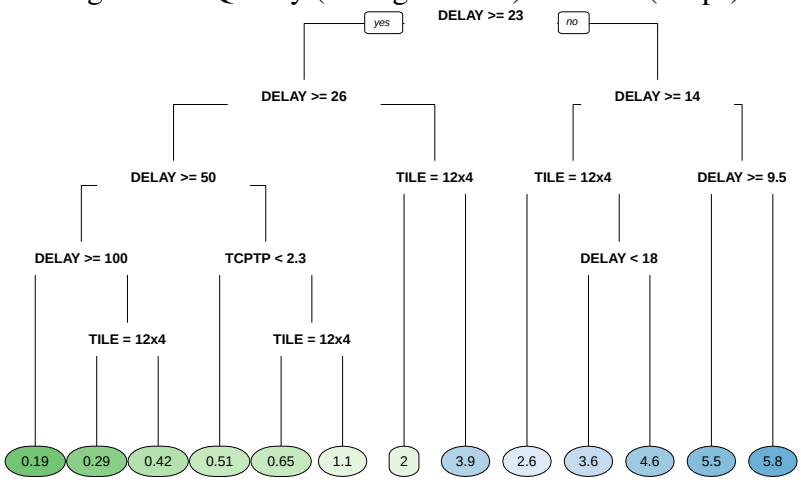
Before assessing the performance of the two-stage method, a thorough analysis of the regression trees was performed. This analysis aims at characterizing the relationship between the input parameters (network conditions and VR video structure) and the VR playout, allowing one to pinpoint to the most influential inputs. Figures 5.6 to 5.13 present the outcome predictors derived from the regression trees. All presented trees share two structural characteristics. First, although inversions may occur, usually the leftmost leaf node holds the lowest value for the predicted variable, and the value increases while moving towards the rightmost leaf node. Second, the closer to the root node, the more important the prediction feature (*i.e.*, delay, TCP throughput, loss and tile scheme).

Having a first look at the content of the trees, two observations can be made. First, network packet losses are not included in any of the trees. This means that the level of packet losses does not have influence on the VR playout performance metrics. Its effect will only be important as they affect the TCP throughput (higher network packet losses = lower TCP throughput). Furthermore, network delays turn out to be the most influential parameter on the VR playout.

Regarding quality (by means of the average bitrate) (Figures 5.6 to 5.8), let us consider the following aspects. The first decision taken in Zone 1, at the root node and, therefore, the most influential, is to understand if the network delay is greater than 23 ms (Figure 5.6). The left branch ($Delay \geq 23ms$) is related to predicted quality not higher than 3.9 Mbps, regardless of any other input value. In other words, even considering that the evaluated LTE network presents TCP throughput of up to 95 Mbps, it is not enough to achieve the maximum bitrate (6.0 Mbps - 4K), if the delay is higher than 23 ms. The reasoning behind this behavior is that each video segment (1 s) demands the download

of 32 (8x4 tiling) or 48 (12x4) tiles. Despite the reuse of the TCP connection avoids the TCP slow-start restart (BLANTON; PAXSON; ALLMAN, 2017)), the request/response overhead limits the throughput.

Figure 5.6: Quality (average bitrate) - Zone 1 (Mbps)



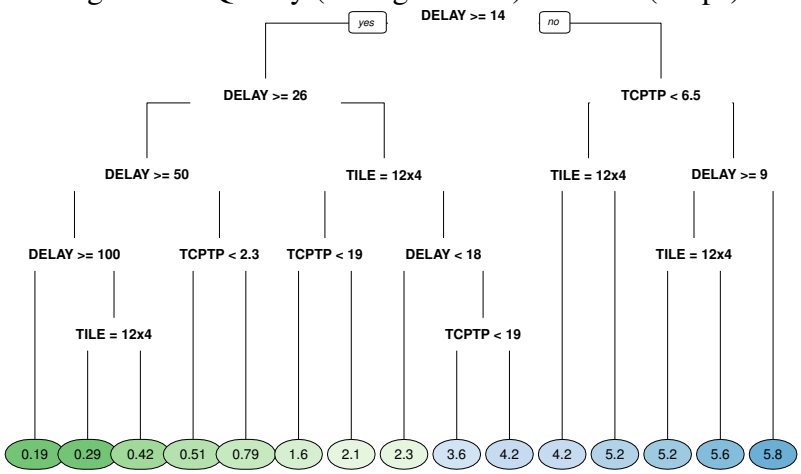
Source: by author (2019).

For Zones 2 and 3 (Figures 5.7 and 5.8), the quality predictors follow a very similar trend. However, in order to achieve the same level of average quality, they demand higher network performance than Zone 1. The right-most leaf of Zones 2 and 3 are a clear example of this behavior. To achieve the same quality (average bitrate of 5.8 Mbps), Zone 2 requires a delay lower than 9 ms, and Zone 3 lower than 7.5 ms. Also, the values of TCP throughput to achieve intermediate average bitrates are higher for Zones 2 and 3 when compared with Zone 1. The main reason for such behavior comes from the rate adaptation heuristic, which prioritizes high bitrates for the tiles that are closest to the viewport's center (Section 5.1.2). Thus, intermediate network performance may be enough to keep Zone 1 at the highest available bitrate, while high levels of network performance allow increasing the bitrate for all zones.

Quality switches (Figures 5.9 to 5.11) provide valuable information in the context of HAS videos. For example, if no switches occur, the full video playout occurs in the lowest available resolution, meaning that the video player is unable to switch to higher bitrates, probably, due to insufficient network performance. In turn, when subject to excellent network performance conditions, most of HAS rate adaptation heuristics (including the one used in this document) will stabilize at the highest available bitrate within a few switches.

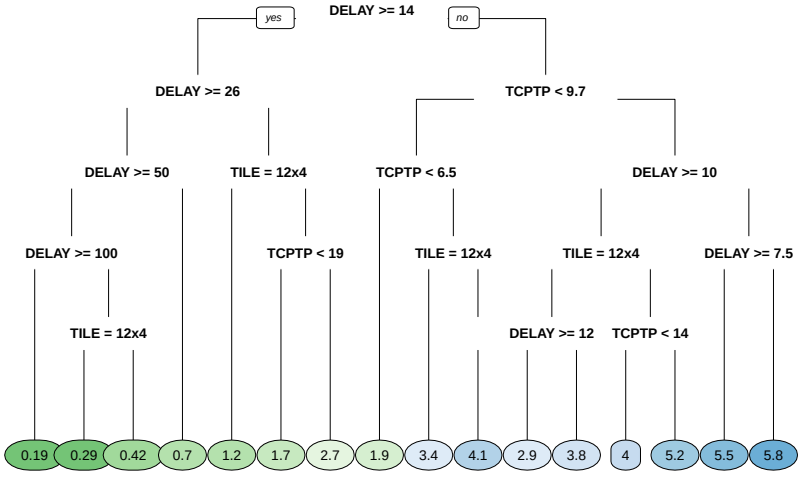
When considering real-world networks, if we have a look at the quality switches

Figure 5.7: Quality (average bitrate) - Zone 2 (Mbps)



Source: by author (2019).

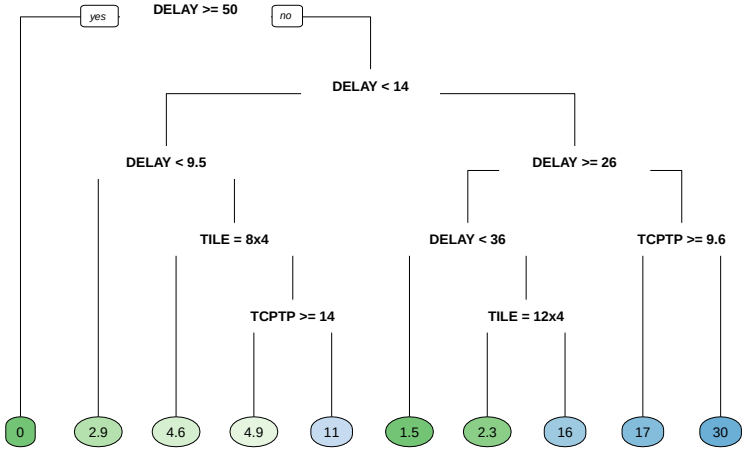
Figure 5.8: Quality (average bitrate) - Zone 3 (Mbps)



Source: by author (2019).

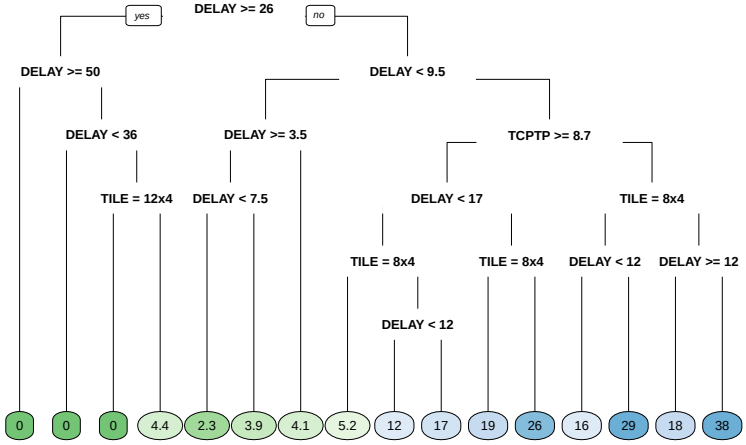
trained trees (Figures 5.9, 5.10 and 5.11), it can be seen that the turning point from zero switches to maximizing the quality is a network delay of 50ms for Zone 1, and 26ms for Zones 2 and 3. However, by analyzing the rightmost leaf nodes of the decision trees for Zones 1, 2 and 3, one can observe that the maximum number of quality switches increases from Zone 1 towards Zone 3: (30, 38 and 58, respectively). This happens because, according to the considered heuristic for rate adaptation, the tiles inside Zone 3 will be the first ones to be switched to a lower resolution in case a network performance degradation is detected, followed by Zone 2 and, if the network performance degradation is severe, the Zone 1.

Figure 5.9: Quality switch - Zone 1



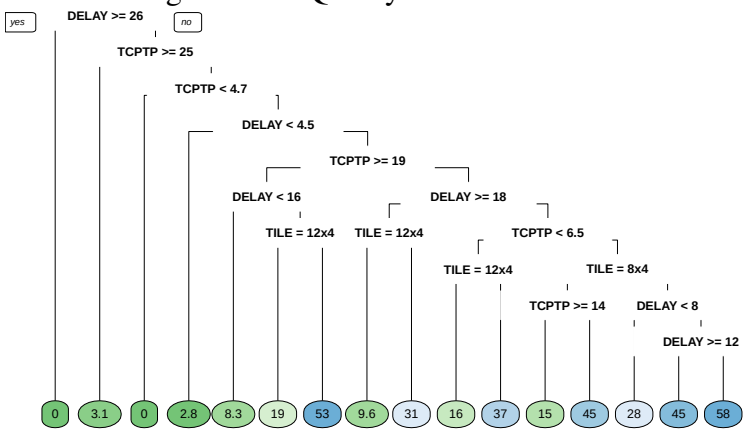
Source: by author (2019).

Figure 5.10: Quality switch - Zone 2



Source: by author (2019).

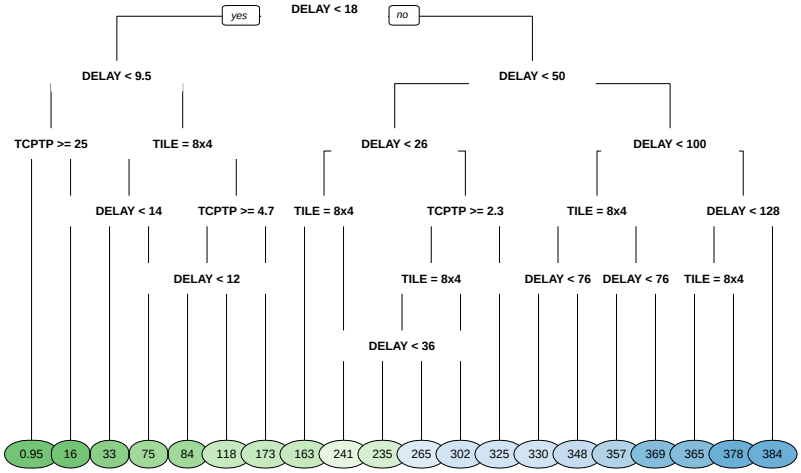
Figure 5.11: Quality switch - Zone 3



Source: by author (2019).

With respect to the cumulative stall time (Figure 5.12), the resulting regression tree presents a wide range of predicted values (from 0.95 up to 384 seconds). One key aspect is related to the decision taken at the root node. As one can observe, if the delay is higher or equal to 18 ms, the minimum expected stall time is equal or higher than 163 seconds, independent of the tiling scheme or the available bandwidth (TCP throughput). Such high values would inflict a dramatic degradation on the perceived quality. In turn, for network delays lower than 9.5 s and TCP throughput equal or higher than 25 Mbps, the expected stall time is minimal (0.95 seconds). It is worth mentioning that, even if the delay is lower than 9.5 s, if the TCP throughput is lower than 25 Mbps, the expected stall time is 16 seconds. Also, in line with the aforementioned findings, the 12x4 tiling scheme leads to a significative higher amount of stall time for intermediate levels of network performance.

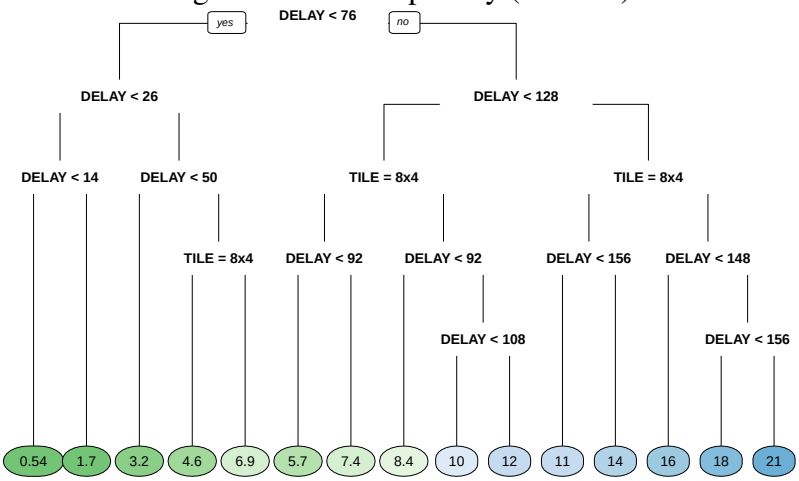
Figure 5.12: Cumulative stall time (seconds)



Source: by author (2019).

Finally, the regression tree for predicting startup delay is shown in Figure 5.13. In the considered VR video player, the startup delay is characterized as the elapsed time between the arrival of the request for the first tile and the completion of the buffer filling for all tiles for the first two segments. As the segment is relatively small, and considering the small file size of the tile chunks (on average 23 KB for 4K video resolution), the startup delay exclusively depends on the network delay. A delay lower than 26 ms is enough to provide an acceptable startup delay (smaller than 1.7 s). However, the best performance is achieved when the delay is lower than 14 ms (0.54 s).

Figure 5.13: Startup delay (seconds)



Source: by author (2019).

5.3.4 PERCEIVE Results

Aiming at determining the accuracy of the proposed predictors, the trained regression trees were used on unseen samples of the generated dataset, according to a 10-fold cross-validation scheme (KOHAVI et al., 1995). We considered as ground truth the performance measured by the reference VR video player when subjected to real-world network performance traces. In light of this, each test sample i contains the predictor variables (*i.e.*, TCP throughput, delay and tiling scheme), and the respective measured values for the performance metrics (*i.e.*, average bitrate, stall time, quality switch and startup delay).

Furthermore, based on the predicted VR playout characteristics, the QoE indexes were estimated by means of Equation 5.2. The parametric constants shown by the model were set to the values presented in Table 5.3. Based on the results shown by Mao et al. (MAO; NETRAVALI; ALIZADEH, 2017), the q function was set to linear, where q is equal to the bitrate. In addition (also according to (MAO; NETRAVALI; ALIZADEH, 2017)), the stall and startup weights (μ and ω) were set to 4.3. The value of the quality switches constant (λ) was tuned to 1 (YIN et al., 2015b). Finally, the zones weights (α_1, α_2 and α_3) were empirically set to 0.7, 0.3 and 0, for Zone 1, Zone 2 and Zone 3. The reason behind setting α_3 to zero comes from the perfect prediction scenario considered in the evaluation. In such cases, the FoV will correspond 100% of tiles of Zones 1 and 2. Thus, there is no influence of the quality of Zone 3 on the user' s perception. In the case that perfect prediction would not be possible, the weights would need to be tuned accordingly.

Table 5.3: PERCEIVE results: constants and function values assigned to the function to estimate QoE (refer to Equation 5.2)

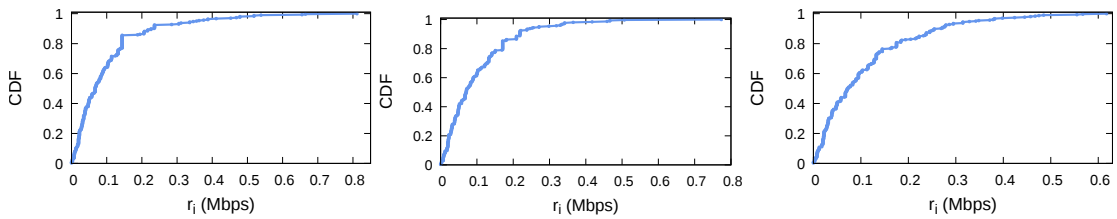
Parameter	Value	Description
q	Linear	Quality function
μ	4.3	Weight for stall time
ω	4.3	Weight for startup delay
λ	1	Weight for quality switches
α_1	0.7	Weight for Zone 1
α_2	0.3	Weight for Zone 2
α_3	0	Weight for Zone 3

Source: by author (2019).

The performance of the method is assessed by means of the residual error calculated between real data sample (entry in the training set) and the predicted one (as already introduced in Section 5.3.1 and Equation 5.3). With the purpose of generalizing the method for videos of arbitrary duration, the residual error for the metrics average bitrate, quality switch and startup delay are normalized by the factor N of the residual error equation, which corresponds to the considered video length (200 seconds). Figures 5.14, 5.15 and 5.16 show the Cumulative Distribution (CDF) of the residual error for the four VR playout performance metrics and the QoE estimation.

Looking at the quality prediction capacities of PERCEIVE (Figures 5.14(a) to 5.14(c)), it is possible to observe that the residual errors are very small (224 Kbps and 220 Kbps for Zones 1 and 2, respectively, for over 90% of the cases). If normalized by the maximum available quality (6.0 Mbps), it represents only 3.73% and 3.67% of residual error. This means that in roughly 97% of the cases, the quality levels are correctly predicted. Even though the residual error for Zone 3 is slightly higher (4.5%), it is still within the acceptability range.

Figure 5.14: Residual error CDFs for quality (average bitrate)

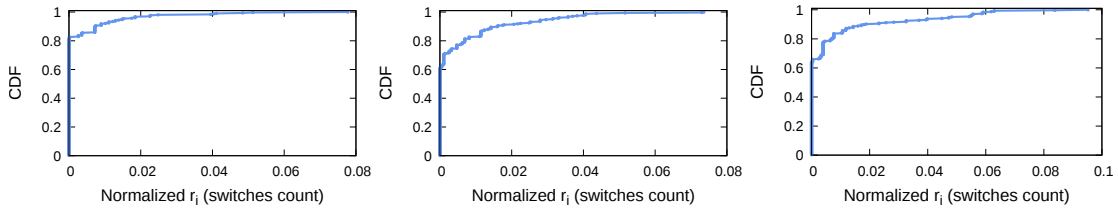


(a) Quality (average bitrate) - Zone 1 (b) Quality (average bitrate) - Zone 2 (c) Quality (average bitrate) - Zone 3

Source: by author (2019).

The accuracy of the quality switch prediction (Figures 5.15(a) to 5.15(c)) shows

Figure 5.15: Residual error CDFs for the quality switch



(a) Quality switch - Zone 1 (b) Quality switch - Zone 2 (c) Quality switch - Zone 3

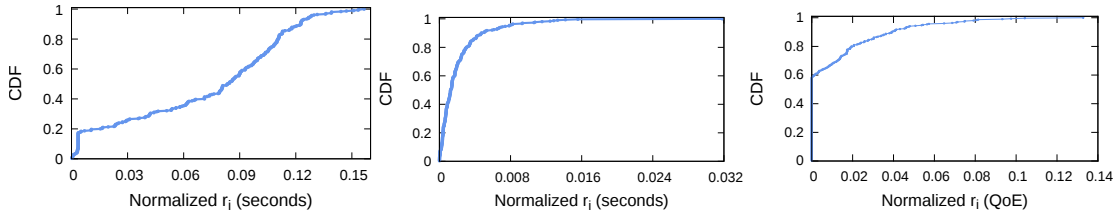
Source: by author (2019).

even better results. For over 90% of the samples, Zones 1, 2 and 3 present a residual error of $r_{i_1} \leq 0.00745$, $r_{i_2} \leq 0.01604$ and $r_{i_3} \leq 0.01877$, respectively. In line with the findings for the average bitrate prediction, Zone 3 presented a higher residual error (1.9%), as this is the zone with the highest number of quality switches during the video playout. In Figure 5.15(b) it is possible to observe that on over 80% of samples the residual error is zero. This is because the quality switch behavior for both extreme cases of the network performance is predictable: first, when the network performance is sufficiently high, the rate adaptation will stabilize at the highest representation, and no further quality switches are expected. Second, when the network performance is degraded, the rate adaptation will keep the video playout at the lowest available quality representation, and, similarly, no further switches are expected.

The stalling time (Figure 5.16(a)) shows an error close to 13% for over 90% of the testing samples. One main reason behind such increased residual error is the wide range of the predicted variable (as we saw in the regression tree of Figure 5.12). Nevertheless, several samples in the training dataset presented zero seconds of stall time. We found that such predictable cases are associated with high levels of network performance. For each of these samples, a residual error of 0.95 was accounted (as 0.95 is the lowest predicted value). As the presented regression tree is the optimal prune, further growth would lead to overfitting, and thus a higher cross-validation error. Due to the relatively high stall time for intermediate and degraded network performance, the prediction performance is impaired as the network performance degrades. However, at high levels of stall time, the QoE is already completely degraded. Thus, the increased error does not impair the accuracy of the QoE estimation.

The final VR playout parameter, the startup delay (Figure 5.16(b)), is characterized as the elapsed time between the request of the video and the playout of the first segment. In the considered context, the startup delay prediction presented a well predictable pattern

Figure 5.16: Residual error CDFs for stall time, startup delay and QoE estimation



(a) Cumulative Stall time (b) Startup delay (c) QoE

Source: by author (2019).

with $r_i \leq 0.00473$ for over 90% of the cases. Also, the regression tree presented a stable prediction performance across all the evaluated samples.

Finally, Figure 5.16(c) depicts the residual error for the QoE estimation. By applying the QoE model defined in Section 5.2.2 to each sample i, it is possible to estimate QoE for both the predicted playout values and the original ones. Then, the residual error can be calculated. Through this procedure, the QoE estimation error induced by the proposed prediction scheme can be assessed. As shown in Figure 5.16(c), the QoE estimation presents $r_i \leq 0.03922$ for over 90% of the cases.

Once a predictive model for the VR video context has been shown to be feasible, the next chapter presents VR-EXP, an experimentation platform that allows for in-depth evaluation of VR video optimization approaches.

6 DISSECTING THE PERFORMANCE OF STATE-OF-THE-ART VR VIDEO OPTIMIZATION TECHNIQUES

In this chapter, we introduce VR-EXP, an open-source platform for carrying out VR video streaming performance evaluation. The platform is capable of systematically evaluating different combinations of VR video streaming optimization approaches. Also, VR-EXP allows pinpointing the interplay between a set of optimization techniques and variable network performance. Furthermore, we consolidate a set of relevant VR video streaming techniques and evaluate them under variable network conditions, contributing to an in-depth understanding of what to expect when different combinations are employed. To the best of our knowledge, this is the first work to propose a systematic approach, accompanied by a software toolkit, which allows one to compare different optimization techniques under the same circumstances[1]. Extensive evaluations carried out using realistic datasets demonstrate that VR-EXP is instrumental in providing valuable insights regarding the interplay between network performance and VR video streaming optimization techniques.

The remainder of this chapter is organized as follows. In Section 6.1, we introduce VR-EXP, encompassing its main components and design choices. In Section 6.2, we outline the evaluation setup including the considered parameters and datasets. Finally, in Section 6.3, we present and discuss the main results.

6.1 VR-EXP - VR Video Experimentation Platform

In this section, we introduce the VR-EXP platform. We start by presenting the general scheme, highlighting its inputs, outputs, and main modules. Next, we introduce the VR video client emulator and its main components. Finally, we propose alternatives for enforcing network performance conditions.

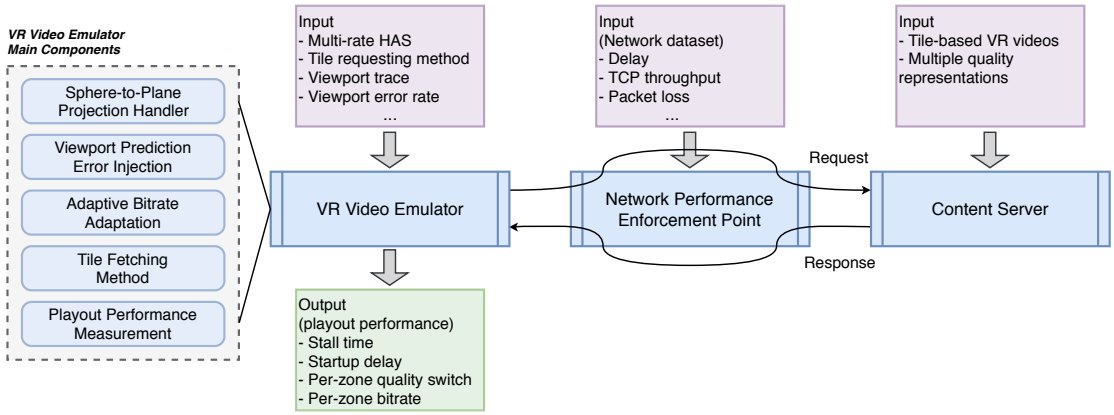
[1]This chapter is based on the following paper:

- Roberto Iraja Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck, Luciano Paschoal Gaspary. **Dissecting the Performance of VR Video Streaming Throughthe VR-EXP Experimentation Platform**. Submitted to the ACM Transactions on Multimedia Computing, Communications, and Applications.

6.1.1 VR-EXP General Scheme

In a nutshell, the VR-EXP platform enables evaluating the interplay between a set of adaptive tile-based VR streaming optimizations and variable network performance conditions. Figure 6.1 depicts the main modules of VR-EXP. The proposed method consists of systematically fetching VR videos through a controlled network environment. From a client perspective, the adaptive VR video client emulator coordinates the use of several VR video techniques upon requesting VR videos from a content server. During the streaming session, the Network Performance Enforcement Point enforces realistic network conditions on the network links between the VR Video Client Emulator and the Content Server. Once the VR video streaming session is finished, VR-EXP reports key VR video playout performance metrics.

Figure 6.1: VR-EXP general scheme.



Source: by author (2019).

The most important module is the VR video client emulator, which is responsible for processing the input parameters, emulating state-of-the-art VR video optimization approaches, and measuring the playout performance. Except for the rendering process, the VR video client emulator mimics the behavior of a VR video adaptive streaming client. Although the rendering task is important for the VR video context, it is mostly related to the HMD rendering capabilities of each device. Therefore, in this work we are interested in evaluating the influence of variable network performance on VR adaptive streaming in an isolated manner, without the interference of HMD particularities. To emulate a dynamic network topology as well as enforcing real-world conditions, VR-EXP relies on either an SDN network controller or the Linux Traffic Controller. In the proposed

platform, adaptive VR videos are delivered by an HTTP server (Apache[2]), which delivers tile-based VR videos in multiple quality representations according to the HAS scheme.

The emulation of the entire VR video streaming ecosystem requires the configuration of several parameters and inputs. For flexibility, VR-EXP enables the definition of its parameters at run time. It allows building scripts for automating complex and extensive experiments. For example, it is possible to parameterize the VR video client emulator by defining behavior characteristics such as the tile requesting method, the rate adaptation heuristic, the expected viewport prediction error, and so forth. In turn, the network module is expected to be fed with a dataset containing a set of network performance metrics (*e.g.*, delay, packet loss rate, TCP throughput). It then enforces these conditions into the emulated links connecting the VR video client emulator to the content server. Once all the input datasets and parameters are configured, the VR video client emulator starts fetching VR videos using the HTTP protocol. After processing the VR video, the emulator writes an output file containing the processed VR video performance metrics, as well as the raw performance data, as described in Section 6.1.2. The complete set of source code and datasets related to the VR-EXP platform are released under GNU General Public License v3.0 and are publicly-available at GitHub[3].

6.1.2 VR Video Client Emulator

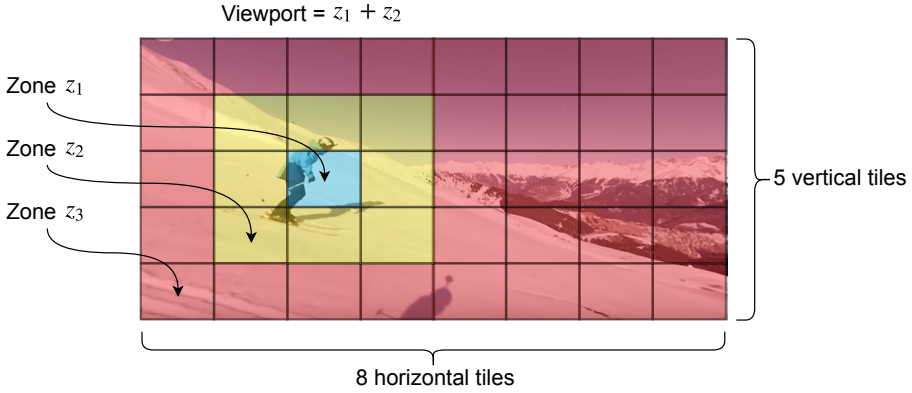
We now focus on the high-level overview of the main functional components of the VR video client emulator[4]. The VR-EXP video client emulator is an extensible and fully parameterized headless VR video client emulator. The source code is written in the C language using the Curl[5] library to systematically fetch tile-based VR videos over the HTTP protocol. The emulator is composed of five main components (Figure 6.1): (*i*) sphere-to-plane projection handling, (*ii*) viewport prediction error injection, (*iii*) bitrate adaptation, (*iv*) tile fetching method, and (*v*) playout performance measurement. Next, we describe their functionality.

Sphere-to-Plane Projection Handling. Several state-of-the-art approaches for VR video streaming optimization rely on tile-based projection schemes (PETRANGELI et al., 2017; GRAF; TIMMERER; MUELLER, 2017; HOU et al., 2018; FILHO et al., 2018). Addi-

[2]Apache HTTP Server: https://httpd.apache.org/
[3]VR-EXP: <https://github.com/rtcostaf/TOMM2019_VR-EXP/>
[4]For additional details, please refer to the documentation available at VR-EXP (VR-EXP, 2019).
[5]Curl: <https://curl.haxx.se/libcurl/c/>

tionally, modern QoE estimation models employ tile clustering methods for manipulating groups of tiles in a coordinated way, depending on their spatial position (FILHO et al., 2018). To cope with these features, VR-EXP is designed to support different tiling schemes and tile clusterization into multiple zones. A multi-zone approach is in line with the notion that the spatial position in which a VR video degradation occurs is vital for estimating QoE. For example, Figure 6.2 depicts an 8x5 tiling scheme which is divided into three zones, where Zone 1 is defined as containing only the viewport's central tile, Zone 2 encompassing the viewport border tiles (8 tiles), and Zone 3 containing the 31 remaining tiles.

Figure 6.2: Example of an 8x5 tiling scheme organized in three zones.



Source: by author (2019).

Viewport Error Injection. Once the projection handler is capable of dealing with several tiling schemes, the next step towards efficient VR streaming consists of emulating the viewport prediction. More precisely, in order to provide an accurate simulation of the entire VR context, the most significative information regarding any heuristic is the viewport prediction error. As discussed in Chapter 2, viewport prediction algorithms present highly variable accuracy depending on many factors. To allow for an accurate evaluation of error patterns, VR-EXP provides a controlled viewport error injection during the streaming session. We designed a flexible viewport error injection component which takes as input viewport traces (*i.e.*, datasets describing the coordinates where the users have looked at in a particular time frame). The viewport trace files contain a full record of coordinates of the VR video, captured at regular intervals (*e.g.*, at each 20ms). In order to provide a mechanism to evaluate the impact of wrong viewport predictions, VR-EXP enables injecting artificial prediction errors when processing the coordinates specified in the trace file. The error injection mechanism can be parameterized with a given error rate, as well as easily extended to support different error models. Using the viewport error injection

can be very helpful in designing novel viewport prediction algorithms. Using VR-EXP, the developer can indeed measure the impact of the error on the resulting performance of the VR sessions. Also, VR-EXP allows understanding what would be the minimum error to guarantee a target performance. The error injection mechanism can be parameterized with a given error rate, as well as easily extended to support different error models. In the current version of VR-EXP, we modeled the viewport prediction error as a random variable with a uniform distribution.

Dynamic Bitrate Adaptation. Taking advantage of the viewport prediction, ABR algorithms provide significative bandwidth savings by selecting appropriate quality representations for each spatial zone. In this procedure, each of the zones is assigned with the most suitable quality level according to both their distance from the center of the viewport and the available bandwidth. VR-EXP currently implements two alternative adaptive streaming heuristics. The general idea of the first heuristic procedure, named Full Delivery (FD) (PETRANGELI et al., 2017), is as follows. Once knowing the available bandwidth in the network (*i.e.*, based on network conditions measured during the download of previous segments), the emulator downloads tiles with the best fit regarding the available bandwidth. For each segment, the heuristic tries to first increase the bitrates on the inner zones of the viewport (Zone Z1 in Figure 6.2). Then, it repeats the procedure to stream tiles from the outer zones (Zones 2 and 3, respectively). Thus, when considering networks with enough available bandwidth, this heuristic will increase the quality representation for all zones. This approach provides effective protection against viewport prediction errors, at the cost of high bandwidth consumption. The second heuristic, named Full Delivery Basic (FDB) (QIAN et al., 2016; FAN et al., 2017), works similarly to the first one. However, instead of increasing the bitrate whenever possible in outer zones, this heuristic increases the bitrates only for zones within the viewport. Although FDB reduces the amount of consumed bandwidth significantly, it may entail QoE degradation in case of viewport prediction errors. Regardless of the approach, the downloaded segments are stored in a playout buffer to be eventually played. Observe that the buffer size plays a significant role in the VR video client performance – particularly regarding viewport prediction accuracy – and, therefore, can be adjusted as needed.

Tile Request Method. On the one hand, the combined use of tile-based VR videos, ABR heuristics, and viewport prediction have proven to be an effective approach to avoid wasting bandwidth. On the other hand, the adaptive tile-based video encoding leads to an increased number of files to be fetched from the content server. For example, consider a

10-minute tile-based VR video, split into 1-second segments, encoded with an 8x5 tiling scheme, and available on three quality representations (*i.e.*, HD, FHD and 4K). For each second of video, it would be necessary to download one quality representation for each tile, which leads to 40 files per second, that is, 24,000 files for a 10-minute streaming session. Considering the above, for each video segment, there is a set of tiles within pre-specified zones to be fetched from the server. VR-EXP allows fetching VR tiles according to two strategies: serial and parallel. On using the serial request method, tiles are fetched from the server, one by one, using multiple (non-parallel) HTTP requests, in a single connection. In turn, in the parallel method, tiles within the same zone (*e.g.,* tiles belonging to the viewport) are fetched in parallel using a configurable number of parallel connections. VR-EXP allows specifying the number of simultaneous connections per zone and splits the set of tiles uniformly among the available connections. It worth mentioning that VR-EXP employs a regular HTTP server (*e.g.*, Apache, NGINX) for hosting the tile-based VR videos, so no specific parameterization is required.

VR Video Playout Performance. To bring together the components detailed throughout this section, along with realistic input datasets, VR-EXP provides a realistic emulation of the VR video streaming ecosystem. Therefore, the next important step toward building a comprehensive VR video evaluation platform is to measure the VR video playout performance accurately. During the video streaming session, VR-EXP assesses a number of VR video playout performance metrics capable of objectively characterizing the quality of the video playout. These metrics include the number of tiles per zone/quality (*e.g.,* number of tiles within the viewport retrieved in 4K resolution), number of quality switches per zone (*i.e.,* number of quality switches on a specific zone), stall time and startup time delay. Table 6.1 provides an example list regarding the playout performance metrics provided by VR-EXP (the complete list may vary depending on the parameterization of the VR-EXP). These metrics were selected because they are the most influential when predicting QoE based on the video streaming playout performance (FILHO et al., 2018). It is worth mentioning that VR video applications rely on TCP/HTTP for providing reliable streaming services. Thus, network performance degradation events, such as packet loss or increased delay, will necessarily translate into either or both quality switches and video stall. Along these lines, VR-EXP focuses on evaluating how multiple VR video optimization techniques interact with variable network performance conditions. Evaluating the distortion introduced by different projection schemes and codecs is out of the scope of this work.

Table 6.1: Playout performance metrics for VR video streaming

Field	Short description	Type/unit
1	Video file name	string
2	User ID (viewport trace ID)	integer
3	Tiling scheme (8x4, 12x4)	string
4	Network trace ID	integer
5	Tile request method (serial, mutli-thread)	integer
6	Network TCP throughput - downlink	Mbps
7	Network delay - RTT	ms
8	Network packet loss - downlink	%
9	Number of 720p tiles - zone 1	integer
10	Number of 1080p tiles - zone 1	integer
11	Number of 4K tiles - zone 1	integer
12	Number of 720p tiles - zone 2	integer
13	Number of 1080p tiles - zone 2	integer
14	Number of 4K tiles - zone 2	integer
15	Number of 720 tiles - zone 3	integer
16	Number of 1080p tiles - zone 3	integer
17	Number of 4K tiles - zone 3	integer
18	Number of quality switches - zone 1	integer
19	Number of quality switches - zone 2	integer
20	Number of quality switches - zone 3	integer
21	Total video stall time	seconds
22	Video startup delay	seconds
23	Average bitrate - zone 1	Mbps
24	Average bitrate - zone 1	Mbps
25	Average bitrate - zone 1	Mbps
26	Viewport error rate (0 - 100)	%

Source: by author (2019).

6.1.3 Network Performance Enforcement

In order to enforce real-world network performance conditions, it is possible to employ, at least, three different strategies: *(i)* network simulation, *(ii)* network emulation or *(iii)* dedicated network infrastructure. The use of network simulation provides great control over the simulated elements. However, simulating the full VR video components stack, plus complex network aspects (*e.g.*, routing, fairness between distinct TCP flavors, operating system features and their limitations) would burden the complexity of implementation and potentially lead to inaccurate simulation results. On the other extreme, dedicated infrastructure provides a realistic environment at the cost of reduced flexibility and complex setup. In light of this, we decided to employ network emulation as we consider this design choice a suitable balance between flexibility and accuracy.

For emulating network links, VR-EXP provides a customized SDN controller (based on Ryu[6]) which, along with Mininet[7], enables reproducing sophisticated network scenarios. The SDN controller is the preferred option for complex network environments due to its ability to easily handle dynamic network topologies and forwarding rules. Also, this strategy allows evaluating the VR video streaming ecosystem when subjected to large

[6]Ryu SDN Controller: <https://osrg.github.io/ryu/>
[7]Mininet: <https://mininet.org>

topologies and high link competition through many concurrent video sessions. However, if the network scenario does not require such complexity (*e.g.*, simulating a few links with static routes), the SDN approach could be replaced with a simpler alternative mechanism (*i.e.*, Traffic Control[8]). Both approaches can benefit from simplified scripting to read input datasets, which describe the network performance (*i.e.*, delay, jitter, residual bandwidth, packet loss) and enforce these network conditions on a target network.

6.2 Evaluation Setup

Using VR-EXP as basis, we carry out an extensive evaluation of state-of-the-art heuristics when subjected to variable network performance conditions. In this section, we present the experimental setup. We start by introducing the 4G/LTE performance dataset, which provides realistic network conditions to the evaluation process. Next, we describe the VR video dataset, including head track traces, which enables the evaluation of viewport-aware approaches. We end this section by outlining the experiment plan and its main procedures.

6.2.1 4G/LTE Performance Dataset

In this work, along with the VR-EXP method and toolkit, we provide a comprehensive dataset for 4G/LTE network performance. The dataset contains the following IP metrics: Round Trip Time (RTT), delay variation (also referred to as jitter), one-way packet loss, and one-way TCP throughput (in the scope of this work also referred as to residual bandwidth). These metrics were gathered by means of IP active measurements, in conformance with the recommendations issued by the IETF IP Performance Metrics Working Group (MORTON, 2016). To obtain these indicators we employed a scalable active measurement-based platform named Netmetric (SANTOS et al., 2007; FILHO et al., 2016; STANGHERLIN et al., 2011). Table 6.2 describes the dataset fields. Except for packet loss, all values are averaged over the whole packet burst.

In Figure 6.3 we present a brief statistical analysis of the measurements available in the dataset regarding the three main metrics. As shown in Figure 6.3(a), the TCP throughput metric presents a wide range of measured values for the downlink. For ex-

[8]TC: <http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html>

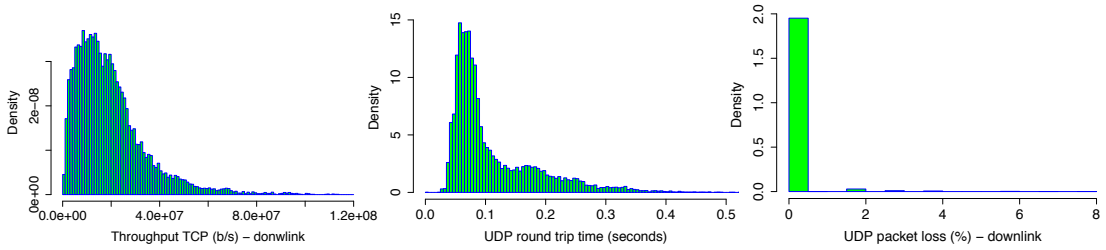
Table 6.2: IP performance dataset (Field 1 is the leftmost column

Field	Short description	Type/unit
1	TCP throughput - uplink	b/s
2	TCP throughput - downlink	b/s
3	Round-trip time (RTT)	seconds
4	Packet loss - downlink	%
5	Packet loss - uplink	%
6	Delay variation (jitter) - uplink	seconds
7	Delay variation (jitter) - downlink	seconds

Source: by author (2019).

ample, the downlink presents a throughput varying from a minimum of 31.4 Kbps to a maximum of 113.2 Mbps, with a median of 16.5 Mbps and a mean of 19.6 Mbps. In turn, Figure 6.3(b) depicts the RTT metric ranging from 1 ms up to 18.5 seconds (the upper limit is not shown in the Figure 6.3(b) due to the long tail), with a median of 81 ms and a mean of 120 ms. Finally, the packet loss (Figure 6.3(c)) for the downlink ranges from 0% up to 8%.

Figure 6.3: Network performance dataset: histograms for TCP throughput, delay and packet loss.



(a) TCP throughput - downlink (b) Delay - RTT (c) Packet loss - downlink

Source: by author (2019).

The network dataset comprises over 14,000 measurements taken from 01/06/2017 to 31/07/2017. Each measurement considers the end-to-end path between the source node, a server located at the premises of the Federal University of Rio Grande do Sul, and a measurement device (destination). The destination of each measurement session is an Android (6.0) smartphone running the measurement agent and attached to a 4G/LTE network. Measurement devices were spread countrywide embracing the four major mobile operators. Together, these operators are responsible for providing mobile services to over 236 million subscribers (TELCO, 2018).

Each measurement session is composed of two bi-directional packet bursts, where the first uses UDP and the second TCP. The UDP packet burst is employed to measure RTT, loss and jitter by injecting 400 packets of 100 bytes at 50 ms intervals. As some operators block the Network Time Protocol (NTP), we decided not to measure the One-Way

Delay (OWD). Instead, the RTT metric was obtained based on a single clock (source). In turn, the TCP burst gauges the TCP throughput for the considered path by injecting 640 packets of 1,488 bytes each. For privacy reasons, sensitive information regarding the considered mobile operators (*e.g.*, operator name, provider ID, cell ID) has been removed from the dataset.

Considering the number of measurements and the wide range of the considered metrics, the network performance dataset may be useful to support further research in several areas. Especially in the field of VR video streaming since the available metrics encompass the network performance indicators that influence video streaming performance the most (*i.e.*, delay and residual bandwidth) (YIN et al., 2015b; FILHO et al., 2018). Additionally, the metrics' ranges allow evaluating high-resolution and tile-based VR videos, including 4K+ resolution. It is worth mentioning that the range for the TCP throughput metric is in line with similar studies conducted in other regions *et al.* (HOOFT et al., 2016).

6.2.2 VR Video Dataset

In this evaluation we use two VR videos from Wu et al.' s dataset (WU et al., 2017), namely "Google Spotlight-HELP" and "Freestyle Skiing". Aiming at evaluating viewport-aware approaches, for each video we also consider the available datasets which describe users' head movements while watching the VR videos. However, the original VR videos are non tile-based, so they needed to be re-encoded. To do so, the first step consisted of extracting the raw YUV files, making use of the Kvazaar encoder (VIITANEN et al., 2015). The resulting encoding produced two tiling schemes: 8×4 and 12×4 (QIAN et al., 2016; KOHAVI et al., 1995). Additionally, each tiling scheme was encoded into three quality representations, namely 720p (1.8Mbps), 1080p (2.7Mbps) and 4K (6Mbps). Next, we employed the MP4Box[9] application to pack the encoded videos into MP4 containers. Then, we sliced each quality representation into 1 second segments. Finally, we used MP4Box to extract per-tile files and to generate the MPEG Dash Media Presentation Description (MPD) files considering multiple quality representations. Table 6.3 summarizes the main parameters regarding the VR video dataset.

[9]MP4Box https://gpac.wp.imt.fr/mp4box/

Table 6.3: Adaptive streaming configurations.

Videos	Qualities (bitrates)	Quality zones	Segment	Tiling
Google Spotlight Freestyle Skiing (Wu et al. (WU et al., 2017))	720p - 1.8Mbps 1080p - 2.7Mbps 4K - 6Mbps	Zone 1: 1 tile (central FoV) Zone 2: 8 tiles (adj. Zone 1) Zone 3: remaining tiles	1 s	12×8 8×4

Source: by author (2019).

6.2.3 Experiment Plan

VR-EXP was deployed on the imec iLab.t Virtual Wall emulation platform[10]. The experiments consisted of employing VR-EXP for measuring VR video performance while subjected to a broad variety of network conditions and multiple VR video optimization techniques. To capture the interplay between the considered variables (detailed in Subsection 6.1.2), we varied the experiment's parameters (*e.g.*, network performance, VR video, tiling scheme, adaptive bitrate heuristic, playout buffer size) in a controlled manner. The experiments were organized around each key VR video optimization technique, namely the viewport prediction error, per tile rate adaptation heuristics and tile requesting method. In a first step, we varied the parameters within each heuristic at a time, assuming default values for the remaining heuristics (according to Table 6.4). In order to capture the interplay within a set of heuristics, in the second step, we carried out a more sophisticated evaluation by varying multiple parameters and heuristics within the same experiment.

VR-EXP is designed to work with any QoE model that supports VR video playout performance indicators as input. Employing a QoE model can be very insightful as it provides a consolidated view regarding the effect of multiple VR video playout performance metrics on QoE. In consonance with state-of-the-art QoE models for traditional video streaming (PETRANGELI et al., 2015; YIN et al., 2015b; MAO; NETRAVALI; ALIZADEH, 2017), we employ a QoE model (FILHO et al., 2018) that is able to translate multiple VR video playout performance characteristics into an estimated QoE score ranging from 1 to 5. To instantiate the QoE model, we consider the three-zone scheme defined by Da Costa Filho *et.al* (FILHO et al., 2018), where Zone 1 refers to the viewport center tile, Zone 2 encompasses the eight tiles surrounding Zone 1, and Zone 3 includes all remaining tiles. We also consider the same constants and function values proposed by the authors, which are summarized as follows (refer to Equations 5.1 and 5.2): $q = Linear$, $\mu = 4.3$, $\omega = 4.3$, $\lambda = 1$, $\alpha_1 = 0.7$, $\alpha_2 = 0.3$, and $\alpha_3 = 0$.

[10]imec iLab.t: <http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html>

Table 6.4: Main VR-EXP input parameters

Parameter	Value/Range	Details
VR video	Google Spotlight-HELP and Freestyle Skiing	Both videos are used in all experiments
Head track traces	Google Spotlight-HELP and Freestyle Skiing	multiple users/head track traces for each video
Video format	MP4 - HEVC tile-based and HAS	Using MP4Box[11]
Video encoder	Kvazaar	Kvazaar encoder (VIITANEN et al., 2015)
HAS	720p (1.8Mbps), 1080p (2.7Mbps) and 4K (6Mbps)	Kvazaar encoder (VIITANEN et al., 2015)
Segment size	1 second	the same for all experiments
Tiling scheme	8x4 and 12x4	Both tiling schemes are used in all experiments
Considered viewport	One central tile and eight border tiles	NA
Viewport error rate	0% up to 100%	Default 0%
Rate adaptation heuristic	FD and BFD	Default BFD
Tile request method	Single thread, 6 threads and 8 threads	Default single thread
Playout buffer	2 sec up to 8 sec	Default 2 sec

Source: by author (2019).

6.3 Evaluation

In this section, we present the results regarding the application of VR-EXP along with the inputs and parameters described in Section 6.2. We start by evaluating the effects of the Viewport Prediction Error (VPE) on VR video playout performance and QoE. Next, we extend this analysis to encompass per tile rate adaptation heuristics, and finally to tile requesting method. We end this section by presenting a more sophisticated scenario, where multiple parameters, heuristics and the network performance conditions vary within the same experiment.

6.3.1 Effects of Viewport Prediction Error

When dealing with traditional 2D video streaming, we use the term video bitrate (*e.g.*, 2 Mbps, 6 Mbps) equivalently with their respective representations of quality (*e.g.*, 1080p, 4K). Also, we can state that there is a correspondence between the average bitrate delivered to the user and the average bitrate that effectively traversed the network (*i.e.*, bandwidth consumption). However, when it comes to tile-based VR video streaming, this relationship becomes less trivial. For example, consider the streaming of a tile-based VR video using a 12x4 tiling scheme and a viewport containing nine tiles. Assume that during most of the streaming session the viewport is displayed in 4K resolution, while the tiles outside the viewport are fetched at 720p. It turns out that the bitrate delivered to the user (visible portion of the VR video) is equivalent to the 4K representation (*i.e.*, 6 Mbps). However, when considering the FDB heuristic for adaptive bitrate, the overall bitrate of the video (*i.e.*, equivalent to the average bandwidth demand during the streaming session)

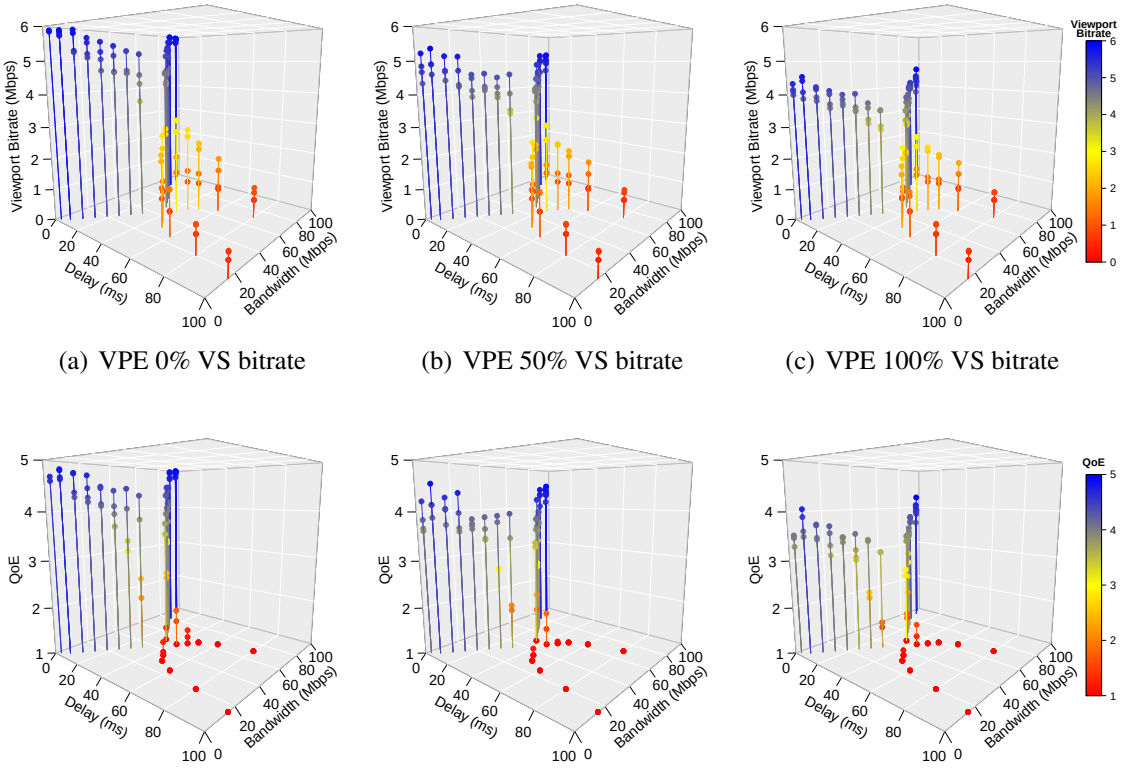
will be slightly higher than the bitrate of the 720p representation. It happens because most of the video (not visible by the user) was fetched in low resolution. For didactic reasons, in this evaluation we use the term *Viewport Bitrate* to denote the bitrate *perceived* by the user, while the term *Video Bitrate* refers to the total bitrate of the video (averaged over all tiles), being equivalent to the bitrate effectively demanded from the network.

As discussed in Chapter 2, depending on the viewport prediction algorithm and the playout buffer size, the viewport prediction accuracy can be quite erratic. In this section, we apply VR-EXP to evaluate the impact of the viewport prediction errors on both video playback performance and QoE. Figure 6.4 shows the performance of the video playout, regarding viewport bitrate and QoE, when subjected to variable network performance conditions and prediction error. Figure 6.4(a) illustrates the baseline scenario, characterized by absence of viewport prediction errors. In this scenario, a network delay below 12 ms is fundamental to provide good levels of viewport bitrate (recall that the bitrate for the 4K representation is 6 Mbps). In such conditions, it is possible to observe viewport rates close to 6 Mbps across a wide range of available bandwidth values.

Figures 6.4(b) and 6.4(c) show how the viewport prediction error affects the viewport average bitrate. When considering a viewport prediction error rate equal to 50% (Figure 6.4(b)), the maximum bitrate decreases approximately by 1 Mbps, while a 100% error in the viewport (Figure 6.4(c)) drops the maximum bitrate to near 4 Mbps, even when considering the most favorable network condition. The viewport error does not affect the playout performance when subjected to significantly degraded levels of network performance (*i.e.*, delay higher than 50 ms). In such cases, the rate adaptation algorithm has no room for increasing the quality representation. All tiles are requested at the lowest available quality representation and, as a direct consequence, a viewport error does not lead to additional degradation. Figures 6.4(d), 6.4(e) and 6.4(f) demonstrate the impact of prediction errors on QoE. One can observe that severe prediction errors (Figure 6.4(f)) may lead to a decrease of up to 2 points in the QoE score when compared to the baseline scenario shown in Figure 6.4(d).

Next, we employed VR-EXP to assess more accurately the effects of the viewport prediction error. To do so we added the tile scheme information. Moreover, we split the rates between the bitrate observed for the tiles within the viewport and the bitrate for the entire video (including the viewport). Figure 6.5(a) shows the baseline case, which considers a perfect viewport prediction. To improve readability, in all plots of Figure 5 we show the network variability only in terms of delay, removing the bandwidth dimension

Figure 6.4: The effects of the viewport prediction error on VR video playout performance and QoE.



(a) VPE 0% VS bitrate (b) VPE 50% VS bitrate (c) VPE 100% VS bitrate

(d) VPE 0% VS QoE (e) VPE 50% VS QoE (f) VPE 100% VS QoE

Source: by author (2019).

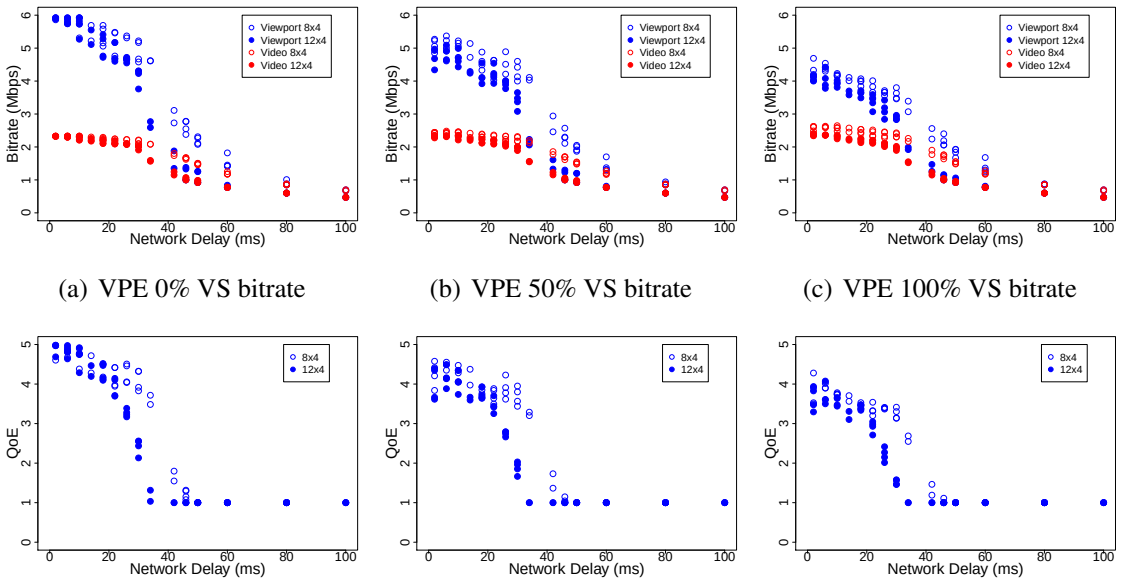
from the analysis. The red dots represent the bitrate for the entire VR video (*i.e.*, viewport + remaining tiles), which is equal to the network bandwidth required for streaming the VR video. When it comes to the viewport (blue dots), both tiling schemes are able to achieve the maximum bitrate when the delay is lower than 12 ms. However, the 8x4 tiling scheme presents significantly better bitrates for intermediate network conditions (delay between 12 ms and 60 ms). This gain is explained by the fact that the HTTP request/response overhead is lower for the 8x4 tiling scheme (32 files per segment) against 48 files per segment for the 12x4 tiling scheme. When the delay is higher than 60 ms, the video playout is totally impaired, and neither the tiling scheme nor the VPE introduces additional degradation.

Complementing the previous analysis, in Figures 6.5(b) and 6.5(c) it is possible to observe that the tiling scheme plays an important role in the video playout performance. The viewport error leads to lower viewport bitrate for intermediate network conditions when compared to the baseline scenario. Still, for intermediate network delay, the 8x4 tiling scheme presents a viewport bitrate up to 2 Mbps higher when compared to the 12x4

tiling scheme. The obtained results indicate that the VPE influences, mainly, the average viewport bitrate and quality switch metrics. The remaining metrics for playout performance (*i.e.*, startup delay and stall time) are not affected by prediction errors. Figures 6.4(d), 6.4(e) and 6.4(f) show that, in line with previous findings, the viewport prediction error has the potential to reduce the QoE score significantly. Nevertheless, the tiling scheme can dramatically influence the QoE score. For example, in Figure 6.5(d) it is possible to observe that, for a network delay of around 35 ms, the 8x4 tiling scheme outperforms the 12x4 by more than 2 points in the expected QoE score.

Main insight for viewport prediction error. Increased levels of VPE may result in reduced viewport quality and QoE. The VPE does not introduce further degradation when subjected to low-performance networks. The tiling scheme has the potential to highly affect QoE when considering intermediate levels of prediction error and network performance.

Figure 6.5: The effects of the viewport prediction error and tiling scheme on playout performance and QoE.



(a) VPE 0% VS bitrate (b) VPE 50% VS bitrate (c) VPE 100% VS bitrate

(d) VPE 0% VS QoE (e) VPE 50% VS QoE (f) VPE 100% VS QoE

Source: by author (2019).

6.3.2 Per Tile Rate Adaptation Heuristics

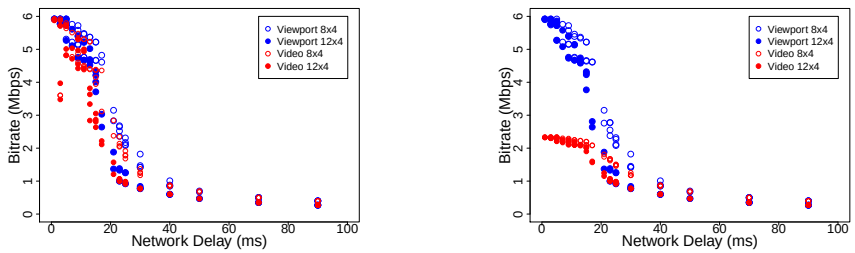
As discussed in Chapter 2, the tile-based rate adaptation algorithm is crucial for achieving a suitable balance between playout performance and network bandwidth con-

sumption. Although VR-EXP can be extended to encompass several strategies, in this section we focus on two distinct approaches, namely the Full Delivery (FD) (PETRANGELI et al., 2017) and the Full Delivery Basic (FDB) (GRAF; TIMMERER; MUELLER, 2017). Recall that both approaches request the tiles inside the viewport in the highest possible quality representation. The main difference between them is that, depending on the available bandwidth, the FD method attempts to increase the bitrate for all the tiles, including the ones outside the viewport. Conversely, the FDB approach does not increase the quality representation for tiles outside the viewport, regardless of the available bandwidth.

Figures 6.6(a) and 6.6(b) show the relationship between the measured viewport bitrate (blue) and the entire video bitrate (red), when subjected to variable network performance conditions. The difference between FD and FDB is more noticeable when the delay is lower than 20 ms. In this case, FD benefits from the available network performance to maximize the quality representation of the entire video. One key advantage of the FD approach is its natural protection against viewport prediction errors, at the cost of increased bandwidth consumption. On the other hand, when considering methods for viewport prediction with low error rates, the FDB method may represent a better choice as it will maintain good levels of QoE while avoiding bandwidth waste. For intermediate network delay (between 20 and 40 ms), both methods perform similarly, because the network performance is sufficient to accommodate only the viewport in high quality. Finally, for a network delay higher than 40 ms, there is no room for increasing the quality representation at all, and both strategies present equivalent performance.

Main insight for rate adaptation heuristics. The FD heuristic provides excellent protection against viewport prediction errors at the cost of increased bandwidth consumption. If combined with low-error viewport prediction algorithms, FDB may potentially lead to reduced bandwidth consumption.

Figure 6.6: Dynamic rate adaptation heuristics: FD and FDB.



(a) Full Delivery (b) Full Delivery Basic

Source: by author (2019).

6.3.3 Multithreaded Tile Downloading

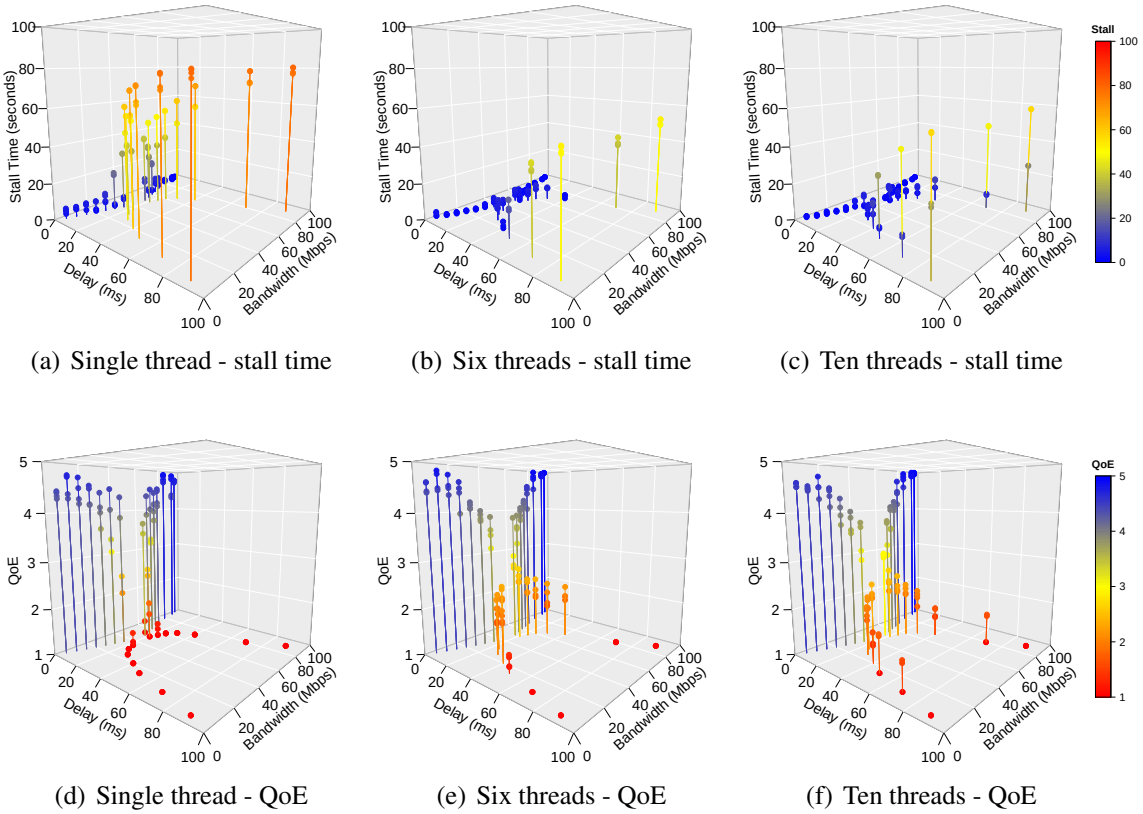
As discussed in Section 6.1, the network delay is the QoS metric that affects video playout performance the most. The reason is that high levels of network delay, when combined with both short video segments and tiling scheme overhead, limit the download throughput. Multithreaded tile request methods can improve the VR video playout performance by reducing the stall time. As shown in Figure 6.7(b), when using six threads it is possible to dramatically reduce the VR video stall time. Basically, when compared to the single thread approach (Figure 6.7(a)), the use of six threads enables handling twice as much network delay (from 20 ms to 40 ms) while maintaining the same level of stall time. When resorting to ten threads for tile downloading (Figure 6.7(c)) it was possible to slightly reduce the stalling time, especially when considering VR videos using the 8x4 tiling scheme (as discussed next).

Figures 6.7(e) and 6.7(f) depict the effects of the multithreaded approach in the QoE score. When compared to the single thread (Figure 6.7(d)), the multithreaded approach is able to increase the QoE score in up to 1.5 points when the delay is higher than 20 ms. However, for network delays higher than 80 ms, the QoE is completely degraded, regardless of the available bandwidth and the use of multithreaded approaches.

Figure 6.8 shows the effect of the multithreaded approach on distinct tiling schemes (*i.e.*, 8x4 and 12x4). When considering a network delay of 40 ms, the six threads approach outperforms the single thread by reducing the stall time from 60 to 5 seconds (approximately) (Figures 6.7(a) and 6.7(b)). The experiment with six threads resulted in similar results for both tiling schemes, with a slight advantage to the 8x4 one. In turn, the ten-thread experiment variation (Figure 6.7(c)) led to an additional reduction of the stall time for the 8x4 scheme, but not for the 12x4, which presented roughly the same results when compared to the six-thread experiment.

Main insight for multithreaded tile downloading. Multithreaded tile fetching can dramatically reduce the stall time and increase the QoE score for intermediate levels of network performance. However, it does not provide noticeable improvements in QoE for either high or low network performance.

Figure 6.7: Multi-thread effect on VR video stall time.

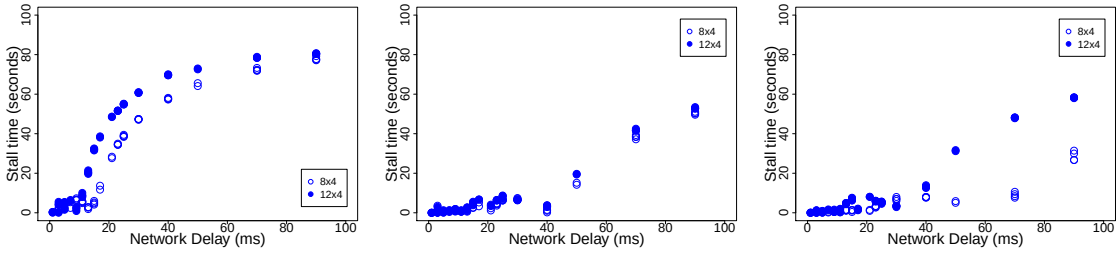


(a) Single thread - stall time (b) Six threads - stall time (c) Ten threads - stall time

(d) Single thread - QoE (e) Six threads - QoE (f) Ten threads - QoE

Source: by author (2019).

Figure 6.8: Multi-thread: stall time and tiling scheme VS network delay.



(a) Single thread - stall time (b) Six threads - stall time (c) Ten threads - stall time

Source: by author (2019).

6.3.4 Buffer Size and Viewport Prediction Error

The evaluation carried out earlier in this section has focused on evaluating the effects of each VR video optimization technique on VR video playout performance and QoE. Aiming to further explore the interplay among different VR video optimization tech-

niques, in this experiment, we evaluate a set of four optimization aspects simultaneously, namely variable viewport scheme, variable viewport prediction error, variable buffer size, and the FDB rate adaptation approach. Additionally, instead of evaluating how the optimization techniques perform when subjected to distinct network performance conditions, in this evaluation we vary the network conditions within the VR video session. Table 6.5 shows ten distinct combinations of network performance indicators that were randomly selected within the range for each QoS metric (as discussed in Section 6.2). A particular VR video session lasts for 60 seconds, where each network performance configuration lasts for 6 seconds, starting with the configuration ID 1 up to the ID 10. The main objective of this experiment is to evaluate the interplay between multiple VR video optimization approaches while subjected to highly variable network performance conditions. To provide a generalized analysis, the results presented in Figure 6.9 represent the averaged values when considering the entire VR video dataset. Therefore, the error bars, in this case, represent the min-max range for each histogram bin.

Table 6.5: Network performance indicators within a 60-second-long VR video session

Conf. ID	Delay (ms)	Bandwidth (Mbps)
1	1	74
2	4	38
3	55	31
4	2	60
5	4	54
6	95	8
7	6	22
8	1	84
9	49	19
10	87	7

Source: by author (2019).

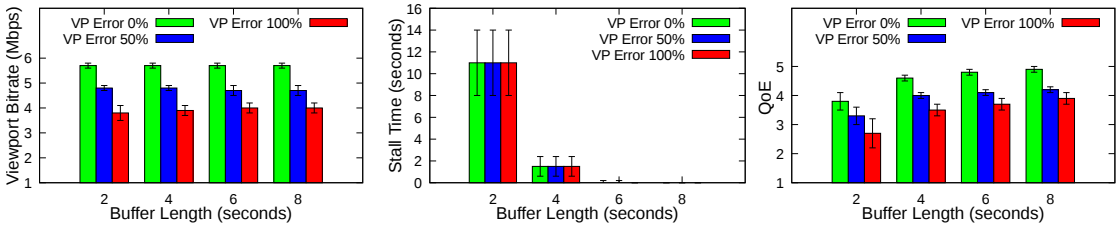
Figure 6.9(a) shows the average quality observed for the viewport when streaming VR videos subjected to variable buffer size and viewport prediction error rates. As discussed in Chapter 2, for most state-of-the-art viewport prediction algorithms, the bigger the buffer size, the higher the prediction error rate. Aiming at evaluating a broad range of scenarios, in the analysis presented in Figure 6.9(a) used a full factorial experiment design considering different values for buffer size and error rate. The obtained results indicate that the viewport prediction error greatly affects the viewport bitrate, while the buffer size itself does not have noticeable influence on it.

Figure 6.9(b) shows that the increased buffer size was able to dramatically reduce the stall time. For example, when considering a playout buffer dimensioned for 4 seconds of video, the stall time drops from 11 seconds to less than 2 seconds (on average). Furthermore, when increasing the buffer to 8 seconds, it was possible to completely eliminate

the stall time. However, as discussed in Chapter 2, most state-of-the-art viewport prediction algorithms experiment sudden accuracy drop when increasing the playout buffer size. Hence, the effective analysis of the interplay between buffer size and viewport error must be done through the evaluation of the QoE indicator, since the QoE score will simultaneously consider both playout performance metrics. Figure 6.9(c) shows that, when using 8 seconds of playout buffer, the worst case scenario for the QoE score (*i.e.*, viewport prediction error of 100%) performs on par with the best case scenario of the 2 seconds buffer (*i.e.*, viewport prediction error of 0%). Furthermore, due to the human randomness, prediction algorithms may present low accuracy even when considering small buffers (*e.g.*, 2 sec). Therefore, using higher values for dimensioning the playout buffer (*e.g.*, 8 sec) will probably outperform smaller buffers setups in most cases.

Main insight for mixed buffer size and prediction error. When dealing with realistic performance levels, increasing the playout buffer size may potentially lead to a better QoE score, even considering the likely increase in the prediction error.

Figure 6.9: The influence of multiple VR video optimization techniques on VR video streaming playout performance and QoE.



(a) Viewport average bitrate (b) Stall time (c) QoE

Source: by author (2019).

7 FINAL CONSIDERATIONS

In this chapter, we present the conclusions about the work carried out in the context of this thesis. Next, we discuss the envisioned future research directions. We conclude this chapter by presenting the achievements obtained during this research.

7.1 Conclusions

The work conducted throughout this thesis suggests that the hypothesis is correct. As discussed next, it provided us with satisfactory answers to the research questions, namely *(i)* would it be possible to employ monitored network indicators to predict playout performance and QoE for both traditional 2D and VR video streaming applications? and *(ii)* How to employ QoE prediction to dynamically select and deploy paths that maximize QoE and minimize infrastructure utilization over time?

As the first piece of work toward building an approach for QoE-aware path selection, we proposed LEAP, a model to predict video streaming playout performance and QoE based on performance indicators of the underlying IP network. To accomplish this objective, the model leverages lightweight active measurements and machine learning techniques (*i.e.*, regression decision trees). The results obtained allowed us to answer the research question 1, as it suggests that it is feasible to estimate application layer performance and QoE for video streaming applications using QoS indicators as predictors. The estimated parameters achieved an average error below 9.92%, and a MOS estimation error below 11% for over 90% of the cases considered. Furthermore, LEAP requires less than 4% of the traffic volume when compared to traditional techniques. The low intrusiveness allows the service provider to configure systematic measurements, with a reduced polling interval, without excessive use of network resources.

While our previous work resulted in a powerful model for estimating QoE for network paths, the model is not scalable for dense networks with thousands of possible paths between a given source and destination. In a second iteration to solve the overall problem of the thesis, we devised a polynomial time complexity QoE-aware path selection algorithm. This building block takes advantage of the previously proposed QoE prediction model and introduces a novel algorithm for performing efficient QoE-aware path selection, in Software-defined Networks, based on per-link QoE composition. In a realistic evaluation involving mobile operator topology and video demand, the proposed approach

outperformed state-of-the-art solutions by impairing at least 37% fewer videos and resulting in four times less stall. Thus, the proposed approach allows answering the research question 2 since it successfully introduces a large-scale QoE-aware deployment approach.

Next, we applied our prediction model to the VR arena by presenting PERCEIVE, a novel performance evaluation method to assess the user's perception of the VR content when streamed through mobile networks. Using machine learning techniques applied to the network performance indicators, it predicts the adaptive VR performance both in terms of playout performance (average viewport bitrate, quality switches, stalling time and starting time) and perceived QoE. To our knowledge, this is the first VR performance model. PERCEIVE has been evaluated considering a real-world environment, in which VR videos are streamed while subjected to an LTE/4G network performance. Then, we assessed its accuracy by means of the residual error between the predicted and measured values. PERCEIVE is able to predict the playout performance metrics with an average prediction error lower than 3.7%, and the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. PERCEIVE not only provides very high prediction accuracy but also allows analyzing the influence of networks on the VR streaming parameters. This feature has helped us pinpoint the network delay as the QoS feature that affects the transport of VR services the most.

The complex interplay between VR video optimization techniques and variable network conditions challenges developers of VR video solutions, as this interaction is neither trivial nor has it been properly investigated. To address this problem, we proposed VR-EXP, an open-source platform for evaluating adaptive VR video streaming that encompasses various optimization techniques and allows for network performance conditions to be varied. Employing VR-EXP, along with realistic datasets, we have produced an extensive assessment that examines the performance of several state-of-the-art optimization techniques when subjected to variable network conditions. The results obtained evidence that the relationship between different optimization techniques for video VR optimization is not trivial. By combining an objective assessment of VR video streaming playout performance and a comprehensive QoE model, VR-EXP allowed pinpointing the components of the VR video ecosystem that most affect the performance of VR video playout and, ultimately, QoE. The benefits of this work are twofold. From the VR video developers' perspective, we expect to contribute a useful approach to conducting a precise and realistic performance evaluation of novel optimization techniques. In turn, from the mobile operator's perspective, we expect VR-EXP to be a valuable tool for supporting

investigations aimed at understanding and predicting how variable network conditions impact VR video performance and QoE delivered to their end-users. Given the above, both PERCEIVE and VR-EXP allowed us to answer research question 1 in the context of the VR video streaming applications.

7.2 Future Research Directions

We envision that this research can be extended in several ways in future investigations. In the context of QoE-aware path selection (for both the traditional 2D and VR video streaming), we believe that an interesting direction is to improve the integration between network-side information (*e.g.*, per link network performance and DPI data) and user's client feedback. For example, it would be possible to employ MPEG's Server and Network Assisted DASH (SAND) (THOMAS, 2015) to provide the network controller with performance data obtained at the end-user premises. The QoE-aware path selection mechanism can be enhanced by considering detailed information regarding the user's traffic, such as the category of the video and CODEC specific information.

When considering the VR video domain, QoE-aware path selection becomes a promising research field. This is due to the fact that the QoE prediction is the first step towards effective QoE-aware path selection. However, most critical components of the VR video ecosystem are still open research questions, thus challenging to predict. In addition to the topics investigated in this thesis, we believe that it would be valuable to examine how advanced aspects of the VR video streaming context, such as eye tracking, saliency detection, and video rendering performance interact with QoE estimation.

Another relevant research direction consists in further exploring the QoE model for VR videos. By comparing the predicted QoE against interviews with end-users, it would be possible to adjust the weights of the prediction model for distinct user sensitivity levels. For example, a group of users can be more sensitive to stall events, while others are more influenced by changes in the quality representation. By conducting large-scale experiments, it would be possible to build users' profiles for the QoE model. The combination of this weight profiling, along with a flexible QoE model, potentially allows for a much finer-grained QoE estimation.

Finally, we intend to investigate approaches aiming at reducing the network overhead imposed by active network performance measurements. Active measurement techniques are still very suitable for mobile networks because they can accurately measure

network performance. Additionally, mobile operators already have tools in place for providing these measurements. However, we deem that emerging developments in data plane programmability, along with In-band Network Telemetry, can provide accurate network performance indicators while reducing the measurement traffic overhead.

7.3 Achievements

The work carried out in the context of this thesis led to the publication of the following peer-reviewed papers:

- **Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks. (Awarded with the ACM Reproducibility Badge)**
  Roberto Iraja Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck , Luciano Paschoal Gaspary.
  ACM Multimedia Systems Conference (MMSys 2018).
- **Scalable QoE-aware Path Selection in SDN-based Mobile Networks. (Best In-session Presentation)**
  Roberto Iraja Tavares da Costa Filho, William Lautenschlager, Nicolas Kagami, Marcelo Caggiani Luizelli, Valter Roesler, Luciano Paschoal Gaspary.
  IEEE International Conference on Computer Communications (INFOCOM 2018).
- **Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE.**
  Roberto Iraja Tavares da Costa Filho, William Lautenschlager, Nicolas Kagami, Valter Roesler, Luciano Paschoal Gaspary.
  IEEE Global Communications Conference (GLOBECOM 2016).

Additionally, the following paper has been submitted to ACM TOMM and is currently under review:

- **Dissecting the Performance of VR Video Streaming Through the VR-EXP Experimentation Platform.**
  Roberto Iraja Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck , Luciano Paschoal Gaspary.
  ACM Transactions on Multimedia Computing, Communications, and Applications.

REFERENCES

AHMAD, A.; FLORIS, A.; ATZORI, L. Qoe-centric service delivery: A collaborative approach among otts and isps. **Computer Networks**, v. 110, p. 168 – 179, 2016. ISSN 1389-1286.

AKHTAR, Z. et al. Oboe: Auto-tuning video abr algorithms to network conditions. In: **Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2018. (SIGCOMM '18), p. 44–58. ISBN 978-1-4503-5567-4. Available from Internet: <http://doi.acm.org/10.1145/3230543.3230558>.

ALMQUIST, M. et al. The prefetch aggressiveness tradeoff in 360° video streaming. In: **Proceedings of the 9th ACM Multimedia Systems Conference**. New York, NY, USA: ACM, 2018. (MMSys '18), p. 258–269. ISBN 978-1-4503-5192-8. Available from Internet: <http://doi.acm.org/10.1145/3204949.3204970>.

ANSARI, R. I. et al. 5g d2d networks: Techniques, challenges, and future prospects. **IEEE Systems Journal**, p. 1–15, 2018. ISSN 1932-8184.

BALACHANDRAN, A. et al. A quest for an internet video quality-of-experience metric. In: **Proceedings of the 11th ACM Workshop on Hot Topics in Networks**. [S.l.: s.n.], 2012. p. 97–102.

BALACHANDRAN, A. et al. Developing a predictive model of quality of experience for internet video. In: **Proceedings of the ACM SIGCOMM 2013**. [S.l.: s.n.], 2013. p. 339–350.

BAO, Y. et al. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In: **2016 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2016. p. 1161–1170.

BLANTON, E.; PAXSON, D. V.; ALLMAN, M. **RFC 5681 - TCP Congestion Control**. 2017. Available at: https://rfc-editor.org/rfc/rfc5681.txt. Accessed on: Feb. 11, 2018.

BOYCE, J. et al. Jvet common test conditions and evaluation procedures for 360 video. **Joint Video Exploration Team of ITU-T SG**, v. 16, 2017.

CASAS, P. et al. An educated guess on qoe in operational networks through large-scale measurements. In: **Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks**. New York, NY, USA: ACM, 2016. (Internet-QoE '16), p. 1–6. ISBN 978-1-4503-4425-8.

CASAS, P.; SCHATZ, R.; HOSSFELD, T. Monitoring youtube qoe: Is your mobile network delivering the right experience to your customers? In: **Proceedings of the Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2013. p. 1609–1614.

CHEN, Z.; LI, Y.; ZHANG, Y. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. **Signal Processing**, v. 146, p. 66 – 78, 2018. ISSN 0165-1684. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0165168418300057>.

CHIKKERUR, S. et al. Objective video quality assessment methods: A classification, review, and performance comparison. **IEEE Transactions on Broadcasting**, v. 57, n. 2, p. 165–182, June 2011. ISSN 0018-9316.

CISCO. **Cisco Visual Networking Index: Forecast and Trends, 2017–2022**. [S.l.], 2018.

CONCOLATO, C. et al. Adaptive streaming of hevc tiled videos using mpeg-dash. **IEEE Transactions on Circuits and Systems for Video Technology**, PP, n. 99, p. 1–1, 2017. ISSN 1051-8215.

CORBILLON, X. et al. Viewport-adaptive navigable 360-degree video delivery. In: **2017 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2017. p. 1–7.

DAHLMAN, E. et al. 5g wireless access: requirements and realization. **IEEE Communications Magazine**, v. 52, n. 12, p. 42–47, December 2014. ISSN 0163-6804.

DIMOPOULOS, G. et al. Measuring video qoe from encrypted traffic. In: **Proceedings of the 2016 Internet Measurement Conference**. New York, NY, USA: ACM, 2016. (IMC '16), p. 513–526. ISBN 978-1-4503-4526-2.

DOBRIJEVIC, O.; SANTL, M.; MATIJASEVIC, M. Ant colony optimization for qoe-centric flow routing in software-defined networks. In: **Proceedings of the 11th International Conference on Network and Service Management (CNSM)**. [S.l.: s.n.], 2015. p. 274–278.

ECKERT, M.; KNOLL, T. M.; SCHLEGEL, F. Advanced mos calculation for network based qoe estimation of tcp streamed video services. In: **Proceedings of the 7th International Conference onSignal Processing and Communication Systems (ICSPCS)**. [S.l.: s.n.], 2013. p. 1–9.

FAN, C.-L. et al. Fixation prediction for 360 video streaming in head-mounted virtual reality. In: **Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video**. New York, NY, USA: ACM, 2017. (NOSSDAV'17), p. 67–72. ISBN 978-1-4503-5003-7.

FARSHAD, A. et al. Leveraging sdn to provide an in-network qoe measurement framework. In: **Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM Workshop)**. [S.l.: s.n.], 2015. p. 239–244.

FILHO, R. I. T. da C. et al. Network fortune cookie: Using network measurements to predict video streaming performance and qoe. In: **2016 IEEE Global Communications Conference (GLOBECOM)**. [S.l.: s.n.], 2016. p. 1–6.

FILHO, R. I. T. da C. et al. Predicting the performance of virtual reality video streaming in mobile networks. In: **Proceedings of the 8th ACM on Multimedia Systems Conference**. New York, NY, USA: ACM, 2018. (MMSys'18), p. 0–0. ISBN 978-1-4503-5002-0.

FRANGOUDIS, P. A.; YALA, L.; KSENTINI, A. Cdn-as-a-service provision over a telecom operator's cloud. **IEEE Transactions on Network and Service Management**, v. 14, n. 3, p. 702–716, Sept 2017. ISSN 1932-4537.

GANGWAL, A. et al. Elba: Efficient layer based routing algorithm in sdn. In: **2016 25th International Conference on Computer Communication and Networks (ICCCN)**. [S.l.: s.n.], 2016. p. 1–7.

GRAF, M.; TIMMERER, C.; MUELLER, C. Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation. In: **Proceedings of the 8th ACM on Multimedia Systems Conference**. New York, NY, USA: ACM, 2017. (MMSys'17), p. 261–271. ISBN 978-1-4503-5002-0. Available from Internet: <http://doi.acm.org/10.1145/3083187.3084016>.

GUÉRIN, R.; ORDA, A. Computing shortest paths for any number of hops. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 10, n. 5, p. 613–620, oct. 2002. ISSN 1063-6692.

HE, J. et al. Favor: Fine-grained video rate adaptation. In: **Proceedings of the 9th ACM Multimedia Systems Conference**. New York, NY, USA: ACM, 2018. (MMSys '18), p. 64–75. ISBN 978-1-4503-5192-8. Available from Internet: <http://doi.acm.org/10.1145/3204949.3204957>.

HOOFT, J. van der et al. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. **IEEE Communications Letters**, IEEE, v. 20, n. 11, p. 2177–2180, 2016.

HOSSEINI, M.; SWAMINATHAN, V. Adaptive 360 vr video streaming: Divide and conquer. In: **2016 IEEE International Symposium on Multimedia (ISM)**. [S.l.: s.n.], 2016. p. 107–110.

HOSSFELD, T. et al. Initial delay vs. interruptions: between the devil and the deep blue sea. In: **Proceedings of the 4th International Workshop on Quality of Multimedia Experience (QoMEX)**. [S.l.: s.n.], 2012. p. 1–6.

HOU, X. et al. Predictive view generation to enable mobile 360-degree and vr experiences. In: **Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network**. New York, NY, USA: ACM, 2018. (VR/AR Network '18), p. 20–26. ISBN 978-1-4503-5913-9. Available from Internet: <http://doi.acm.org/10.1145/3229625.3229629>.

HRISTOVA, H. et al. Heterogeneous spatial quality for omnidirectional video. In: **2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)**. [S.l.: s.n.], 2018. p. 1–6. ISSN 2473-3628.

HSU, W. H.; LO, C. H. Qos/qoe mapping and adjustment model in the cloud-based multimedia infrastructure. **IEEE Systems Journal**, v. 8, n. 1, p. 247–255, March 2014. ISSN 1932-8184.

HUQ, K. M. S. et al. Enhanced c-ran using d2d network. **IEEE Communications Magazine**, v. 55, n. 3, p. 100–107, March 2017. ISSN 0163-6804.

ITU. **Measuring the Information Society**. [S.l.], 2016.

ITU. Measuring the information society. 2017. ISSN ISBN 978-92-61-24521-4.

ITU-T. **Methods for subjective determination of transmission quality - P.800**. Geneva, 1998.

JEONG, E. et al. Viewport prediction method of 360 vr video using sound localization information. In: **2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)**. [S.l.: s.n.], 2018. p. 679–681. ISSN 2165-8536.

JIANG, J. et al. Cfa: A practical prediction system for video qoe optimization. In: **13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)**. Santa Clara, CA: USENIX Association, 2016. p. 137–150. ISBN 978-1-931971-29-4.

JIN, X. et al. Qos routing design for adaptive streaming in software defined network. In: **2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)**. [S.l.: s.n.], 2016. p. 1–6.

JULURI, P.; TAMARAPALLI, V.; MEDHI, D. Measurement of quality of experience of video-on-demand services: A survey. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 401–418, Firstquarter 2016. ISSN 1553-877X.

KALITAY, H. K.; NAMBIAR, M. K. Designing wanem: A wide area network emulator tool. In: **Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS)**. [S.l.: s.n.], 2011. p. 1–4.

KATSARAKIS, M. et al. On user-centric tools for qoe-based recommendation and real-time analysis of large-scale markets. **IEEE Communications Magazine**, IEEE, v. 52, n. 9, p. 37–43, 2014.

KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: STANFORD, CA. **Ijcai**. [S.l.], 1995. v. 14, n. 2, p. 1137–1145.

KUIPERS, F. et al. An overview of constraint-based path selection algorithms for qos routing. **IEEE Communications Magazine**, v. 40, n. 12, p. 50–55, Dec 2002. ISSN 0163-6804.

LEIGHTON, T. et al. Fast approximation algorithms for multicommodity flow problems. **Journal of Computer and System Sciences**, v. 50, n. 2, p. 228 – 243, 1995. ISSN 0022-0000.

LIOTOU, E. et al. Shaping qoe in the 5g ecosystem. In: **Proceedings of the Seventh International Workshop on Quality of Multimedia Experience (QoMEX)**. [S.l.: s.n.], 2015. p. 1–6.

MA, L. et al. Buffer control in vr video transmission over mmt system. In: **2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)**. [S.l.: s.n.], 2018. p. 1–5. ISSN 2155-5052.

MAALLAWI, R. et al. A comprehensive survey on offload techniques and management in wireless access and core networks. **IEEE Communications Surveys Tutorials**, v. 17, n. 3, p. 1582–1604, thirdquarter 2015. ISSN 1553-877X.

MAO, H.; NETRAVALI, R.; ALIZADEH, M. Neural adaptive video streaming with pensieve. In: **Proceedings of the Conference of the ACM Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2017. (SIGCOMM '17), p. 197–210. ISBN 978-1-4503-4653-5.

MORTON, A. **RFC 7799 - Active and Passive Metrics and Methods (with Hybrid Types In-Between)**. [S.l.], 2016.

MSAKNI, H.; YOUSSEF, H. Is qoe estimation based on qos parameters sufficient for video quality assessment? In: **Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC)**. [S.l.: s.n.], 2013. p. 538–544.

MURTHY, S. K. Automatic construction of decision trees from data: A multi-disciplinary survey. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 2, n. 4, p. 345–389, dec. 1998. ISSN 1384-5810.

NAM, H. et al. Towards qoe-aware video streaming using sdn. In: **Proceedings of the IEEE Global Communications Conference (Globecom)**. [S.l.: s.n.], 2014. p. 1317–1322. ISSN 1930-529X.

NAM, H.; KIM, K. H.; SCHULZRINNE, H. Qoe matters more than qos: Why people stop watching cat videos. In: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [S.l.: s.n.], 2016. p. 1–9.

NGUYEN, T. C.; YUN, J. Predictive tile selection for 360-degree vr video streaming in bandwidth-limited networks. **IEEE Communications Letters**, v. 22, n. 9, p. 1858–1861, Sept 2018. ISSN 1089-7798.

NIAMUT, O. A. et al. Mpeg dash srd: Spatial relationship description. In: **Proceedings of the 7th International Conference on Multimedia Systems**. New York, NY, USA: ACM, 2016. (MMSys '16), p. 5:1–5:8. ISBN 978-1-4503-4297-1.

PAUDYAL, P.; BATTISTI, F.; CARLI, M. Impact of video content and transmission impairments on quality of experience. **Multimedia Tools and Applications**, v. 2016.

PESSEMIER, T. D. et al. Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching. **IEEE Transactions on Broadcasting**, v. 59, n. 1, p. 47–61, 2013.

PETRANGELI, S. et al. Qoe-driven rate adaptation heuristic for fair adaptive video streaming. **ACM Trans. Multimedia Comput. Commun. Appl.**, ACM, New York, NY, USA, v. 12, n. 2, p. 28:1–28:24, oct. 2015. ISSN 1551-6857.

PETRANGELI, S. et al. An http/2-based adaptive streaming framework for 360 virtual reality videos. In: **Proceedings of the 2017 ACM on Multimedia Conference**. New York, NY, USA: ACM, 2017. (MM '17), p. 306–314. ISBN 978-1-4503-4906-2.

POULARAKIS, K. et al. One step at a time: Optimizing sdn upgrades in isp networks. In: **Proceedings of IEEE INFOCOM**. [S.l.: s.n.], 2017.

QIAN, F. et al. Optimizing 360 video delivery over cellular networks. In: **Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges**. New York, NY, USA: ACM, 2016. (ATC '16), p. 1–6. ISBN 978-1-4503-4249-0.

RAMAKRISHNAN, S. et al. Sdn based qoe optimization for http-based adaptive video streaming. In: **Proceedings of the IEEE International Symposium on Multimedia (ISM)**. [S.l.: s.n.], 2015. p. 120–123.

SAHHAF, S. et al. Adaptive and reliable multipath provisioning for media transfer in sdn-based overlay networks. **Computer Communications**, v. 106, p. 107 – 116, 2017. ISSN 0140-3664.

SANTOS, G. L. dos et al. Uama: a unified architecture for active measurements in ip networks; end-to-end objetive quality indicators. In: **Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM)**. [S.l.: s.n.], 2007. p. 246–253.

SCHLINKER, B. et al. Engineering egress with edge fabric. In: **Proceedings of the 2017 ACM SIGCOMM Conference**. New York, NY, USA: ACM, 2017.

SEUFERT, M. et al. A survey on quality of experience of http adaptive streaming. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 1, p. 469–492, 2014.

SHAFIQ, M. Z. et al. Understanding the impact of network dynamics on mobile video user engagement. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 42, n. 1, p. 367–379, jun. 2014. ISSN 0163-5999.

SODAGAR, I. The mpeg-dash standard for multimedia streaming over the internet. **IEEE MultiMedia**, v. 18, p. 62–67, 11 2011. ISSN 1070-986X. Available from Internet: <doi.ieeecomputersociety.org/10.1109/MMUL.2011.71>.

SPITERI, K.; SITARAMAN, R.; SPARACIO, D. From theory to practice: Improving bitrate adaptation in the dash reference player. In: **Proceedings of the 9th ACM Multimedia Systems Conference**. New York, NY, USA: ACM, 2018. (MMSys '18), p. 123–137. ISBN 978-1-4503-5192-8. Available from Internet: <http://doi.acm.org/10.1145/3204949.3204953>.

SPITERI, K.; URGAONKAR, R.; SITARAMAN, R. K. Bola: Near-optimal bitrate adaptation for online videos. In: **IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications**. [S.l.: s.n.], 2016. p. 1–9.

STANGHERLIN, K. et al. One-way delay measurement in wired and wireless mobile full-mesh networks. In: **2011 IEEE Wireless Communications and Networking Conference**. [S.l.: s.n.], 2011. p. 1044–1049. ISSN 1525-3511.

TELCO. **Mobile Operators Market Share in Brazil**. 2018. Accessed 14-January-2018. Available from Internet: <http://www.teleco.com.br/en/en_mshare.asp>.

THOMAS, E. Enhancing mpeg dash performance via server and network assistance. **IET Conference Proceedings**, Institution of Engineering and Technology, p. 8 .–8 .(1), January 2015. Available from Internet: <https://digital-library.theiet.org/content/conferences/10.1049/ibc.2015.0014>.

TOMOVIC, S.; RADUSINOVIC, I.; PRASAD, N. Performance comparison of qos routing algorithms applicable to large-scale sdn networks. In: **IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)**. [S.l.: s.n.], 2015. p. 1–6.

UZAKGIDER, T.; CETINKAYA, C.; SAYIT, M. Learning-based approach for layered adaptive video streaming over sdn. **Computer Networks**, Elsevier North-Holland, Inc., New York, NY, USA, v. 92, n. P2, p. 357–368, dec. 2015. ISSN 1389-1286.

VEGA, M. T.; PERRA, C.; LIOTTA, A. Resilience of Video Streaming Services to Network Impairments. **IEEE Transactions on Broadcasting**, 2018.

VIITANEN, M. et al. Kvazaar hevc encoder for efficient intra coding. In: **2015 IEEE International Symposium on Circuits and Systems (ISCAS)**. [S.l.: s.n.], 2015. p. 1662–1665. ISSN 0271-4302.

VR-EXP. **VR Video Streaming Experimentation Platform**. 2019. Accessed 5-January-2019. Available from Internet: <https://github.com/rtcostaf/TOMM2019_VR-EXP>.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, April 2004. ISSN 1057-7149.

WANG, Z.; CROWCROFT, J. Quality-of-service routing for supporting multimedia applications. **IEEE Journal on Selected Areas in Communications**, v. 14, n. 7, p. 1228–1234, Sep 1996. ISSN 0733-8716.

WINKLER, S.; MOHANDAS, P. The evolution of video quality measurement: From psnr to hybrid metrics. **IEEE Transactions on Broadcasting**, v. 54, n. 3, p. 660–668, Sept 2008. ISSN 0018-9316.

WU, C. et al. A dataset for exploring user behaviors in vr spherical video streaming. In: **Proceedings of the 8th ACM on Multimedia Systems Conference**. New York, NY, USA: ACM, 2017. (MMSys'17), p. 193–198. ISBN 978-1-4503-5002-0.

XIAO, Z. et al. Modeling streaming qoe in wireless networks with large-scale measurement of user behavior. In: **2015 IEEE Global Communications Conference (GLOBECOM)**. [S.l.: s.n.], 2015. p. 1–6.

YIN, X. et al. A control-theoretic approach for dynamic adaptive video streaming over http. In: **Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2015. (SIGCOMM '15), p. 325–338. ISBN 978-1-4503-3542-3. Available from Internet: <http://doi.acm.org/10.1145/2785956.2787486>.

YIN, X. et al. A control-theoretic approach for dynamic adaptive video streaming over http. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 45, n. 4, p. 325–338, aug. 2015. ISSN 0146-4833.

ZHANG, B.; HAO, J.; MOUFTAH, H. T. Bidirectional multi-constrained routing algorithms. **IEEE Transactions on Computers**, v. 63, n. 9, p. 2174–2186, Sept 2014. ISSN 0018-9340.

ZHOU, C. et al. On the effectiveness of offset projections for 360-degree video streaming. **ACM Trans. Multimedia Comput. Commun. Appl.**, ACM, New York, NY, USA, v. 14, n. 3s, p. 62:1–62:24, jun. 2018. ISSN 1551-6857. Available from Internet: <http://doi.acm.org/10.1145/3209660>.

APPENDIX A — SUMMARY IN PORTUGUESE

A.1 Contexto e Motivação

As redes móveis serão responsáveis por mais de dois terços de todo o tráfego IP até 2020 (ITU, 2017). Nesse contexto, as operadoras móveis estão sendo desafiadas pelo tráfego de vídeo, que está pressionando sua infraestrutura de rede para o limite (MAALLAWI et al., 2015). Segundo a Cisco, o tráfego proveniente das aplicações de vídeo foi responsável por 60% do tráfego total da Internet em 2016. E há mais por vir: o tráfego proveniente das aplicações de vídeo deve aumentar 9 vezes até 2022, contabilizando 78% do tráfego total de dados (CISCO, 2018). De acordo com a mesma fonte, os vídeos de Realidade Virtual (VR) aumentarão significativamente esse desafio, pois o tráfego gerado por esse conteúdo deverá aumentar 12 vezes até 2022 (CISCO, 2018). Um importante habilitador para tal crescimento é a difusão de *Head Mounted Displays* (HMDs). Os HMDs estão apresentando altas taxas de penetração, já que eles (*i*) estão se tornando eficazes e acessíveis (p. ex., Google Cardboard/Daydream [1], (*ii*) são fornecidos gratuitamente juntamente com determinados *smartphones* (p. ex., Google Pixel e Samsung Galaxy S) e (*iii*) estão sendo priorizados pela indústria (p. ex., O Facebook anunciou recentemente uma queda permanente no preço do Oculus Go com o objetivo de atingir 1 bilhão de usuários de VR)[2].

As aplicações de streaming de vídeo em realidade virtual são desafiadoras pois: (*i*) vídeos VR serão transportados prioritariamente por redes móveis, já que os dispositivos móveis contabilizarão 71% do tráfego total da Internet até 2022 (CISCO, 2018); (*ii*) redes móveis são caracterizadas por níveis altamente variáveis de desempenho (FILHO et al., 2016); e (*iii*) aplicações de vídeo VR requerem altos níveis de desempenho de rede para prover uma Qualidade de Experiência (QoE) satisfatória (CISCO, 2018). Para fornecer uma noção de como essas aplicações são exigentes, estudos recentes mostraram que, para fornecer níveis adequados de QoE, as aplicações de vídeo de realidade virtual exigem um atraso de rede menor que 9 ms (FILHO et al., 2018), enquanto a largura de banda necessária para transportar vídeos VR de alta resolução pode alcançar 500 Mbps (CISCO, 2018). Nesse nível de demanda, não somente as operadoras terão dificuldades para fornecer serviços com boa relação custo-benefício, mas também os desenvolvedores de conteúdo de vídeo VR também serão desafiados por terem que lidar com aplicações

[1] Google VR: https://vr.google.com/

[2] Um bilhão de pessoas em VR: https://goo.gl/2LNuAo

altamente exigentes.

Para lidar com o enorme crescimento do tráfego de dados, as operadoras móveis necessitam investir constantemente (CAPEX e OPEX) para ampliação de capacidade, atualização de tecnologia (p. ex., 3G, 4G e 5G), bem como para melhorar a cobertura externa e interna. Na direção oposta, a receita média por usuário (ARPU) para banda larga móvel caiu de US$ 23,00 em 2013 para US$ 13,00 em 2015 (ITU, 2016). Todos esses elementos juntos exercem uma enorme pressão sobre as operadoras de rede para que essas gerenciem sua infraestrutura da maneira mais eficiente possível (MAALLAWI et al., 2015).

Com o objetivo de aumentar a eficiência de suas redes, as operadoras móveis contam com tecnologias de descarregamento, tais como *Small Cells* (Femtocell, Picocell), descarregamento de Wi-Fi, redes de distribuição de conteúdo (CDNs) e, em um futuro próximo, comunicação 5G dispositivo a dispositivo (D2D) e *Mobile Edge Computing* (HUQ et al., 2017; ANSARI et al., 2018; FRANGOUDIS; YALA; KSENTINI, 2017). Essas tecnologias são capazes de descarregar diferentes segmentos da rede (ou seja, borda, agregação, núcleo e *peering*), desempenhando um papel fundamental na otimização da infraestrutura de rede. Essas tecnologias permitem encurtar a distância entre o assinante e o conteúdo acessado, evitando o congestionamento da rede, espalhando o tráfego por caminhos alternativos. Como uma indicação da importância dessas tecnologias de descarregamento, somente em 2016, 60% do tráfego de dados móveis foram realocados para caminhos alternativos, considerando apenas o descarregamento de Wi-Fi e Femtocell (CISCO, 2018).

A.2 Definição do Problema

A adoção de técnicas de descarregamento introduz uma grande diversidade de caminhos possíveis pelos quais o tráfego de usuários pode ser escoado e, como consequência imediata, aumenta a complexidade do gerenciamento de rede. Embora muito importante, esse avançado recurso não se traduz diretamente em melhores níveis de QoE (SCHLINKER et al., 2017). Isso é notavelmente verdadeiro se considerarmos que algumas técnicas de descarregamento podem contar com infraestrutura compartilhada e de terceiros, o que possivelmente exacerbaria a imprevisibilidade em relação à QoE fornecida. Portanto, uma tarefa desafiadora para as operadoras de telefonia móvel consiste em como lidar com o crescente tráfego do usuário final, otimizando a utilização da sua infraestru-

tura e gerenciando a QoE do usuário. De fato, do ponto de vista da operadora, enfrentar esse desafio é crucial para se manter competitiva, uma vez que o gerenciamento efetivo da interação entre a QoE percebida pelo assinante e os investimentos em infraestrutura é o principal fator para aumentar o retorno do investimento (NAM; KIM; SCHULZRINNE, 2016; AHMAD; FLORIS; ATZORI, 2016).

Dado o contexto acima, o principal desafio de pesquisa que investigamos ao longo dessa Tese consiste em como aproveitar a diversidade de caminhos introduzidos pelas tecnologias de descarregamento atuais, e também as vindouras (p. ex., 5G D2D, Edge Computing e Fog Computing), para selecionar dinamicamente caminhos capazes de maximizar QoE e minimizar os custos de infraestrutura de rede. Em suma, a tarefa de seleção de caminhos cientes de QoE pode ser decomposta em dois problemas principais. O primeiro problema consiste em prever oportunamente a QoE para caminhos de rede disponíveis. Por sua vez, o segundo problema engloba o algoritmo de seleção de caminho em larga escala, que deve considerar restrições ao selecionar caminhos de rede otimizados (p. ex., objetivo de QoE e recursos de rede disponíveis). O primeiro problema é complexo porque as informações que estão intimamente relacionadas à QoE (p. ex., avaliações subjetivas e medições objetivas) não são amplamente disponíveis ou viáveis para se obter uma abordagem sistemática para redes de larga escala. Em relação ao segundo problema, ele pode ser facilmente resolvido quando considerando pequenas instâncias. No entanto, torna-se notavelmente complexo quando combinado com restrições adicionais (p. ex., utilização de recursos) e aplicado a redes ultra-densas.

A.3 Principais Resultados

Como primeira contribuição dessa Tese, no sentido de construir uma abordagem para seleção de caminhos cientes de QoE, introduzimos LEAP. O modelo proposto é capaz de prever o desempenho da reprodução de streaming de vídeo, bem como seu respectivo QoE, utilizando como entrada indicadores de desempenho da rede IP subjacente. Para atingir esse objetivo, o modelo proposto utiliza medições ativas leves e técnicas de aprendizado de máquina, no caso, árvores de decisão de regressão. Os resultados obtidos sugerem que é possível estimar o desempenho da camada de aplicação e QoE para aplicações de streaming de vídeo usando indicadores de QoS como preditores. Os indicadores estimados alcançaram um erro médio abaixo de 9,92%, e um erro de estimação MOS abaixo de 11%, para mais de 90% dos casos considerados. Além disso, LEAP re-

quer menos de 4% do volume de tráfego quando comparado às técnicas tradicionais. A baixa intrusividade permite que o provedor de serviços configure medições sistemáticas, com um intervalo de medição reduzido, sem resultar em um uso excessivo de recursos de rede.

Embora nosso trabalho anterior tenha resultado em um poderoso modelo para predição de QoE para caminhos de rede, tal modelo não é escalável para redes densas contendo milhares de caminhos possíveis entre uma determinada origem e destino. Em uma segunda iteração sobre o problema central da Tese, propusemos SQAPE, um algoritmo de tempo polinomial para seleção de caminhos cientes de QoE. A abordagem proposta tira proveito do modelo de predição de QoE apresentado anteriormente (LEAP) e introduz uma estratégia inovadora para seleção eficiente de caminhos cientes de QoE em redes definidas por software. Em uma avaliação realista, envolvendo demandas de serviço e topologias reais de operadora móvel, SQAPE superou soluções do estado da arte, resultando em um menor número de vídeos degradados (pelo menos 37%) e acumulando quatro vezes menos congelamentos de imagem.

Aproveitando as lições aprendidas com o modelo de predição de QoE para streaming de vídeo introduzimos PERCEIVE, um modelo para predição de QoE no contexto de vídeos VR. Utilizando técnicas de aprendizado de máquina, o modelo prevê o desempenho das aplicações de vídeos VR, tanto em termos de desempenho de reprodução (bitrate, interrupções de reprodução, chaveamentos de qualidade e atraso para início da reprodução), quanto o seu respectivo QoE. No melhor do nosso conhecimento, este é o primeiro modelo de predição de QoE para vídeos VR. PERCEIVE foi avaliado considerando um ambiente do mundo real, no qual os vídeos VR são transmitidos através de links que emulam o desempenho da rede LTE/4G. Nesse contexto, sua acurácia foi avaliada por meio do erro residual entre os valores preditos e efetivamente medidos. PERCEIVE foi capaz de prever os indicadores de desempenho de reprodução com um erro médio inferior a 3,7%, enquanto a predição de QoE apresentou um erro menor que 4%, para mais de 90% dos casos considerados. PERCEIVE não apenas forneceu uma precisão alta, mas também permitiu analisar a influência dos indicadores de QoS no desempenho das aplicações de vídeo VR.

Em uma última interação sobre o problema central da Tese propusemos VR-EXP, uma plataforma de código aberto para avaliação em profundidade de técnicas de otimização para aplicações de vídeo VR. Empregando VR-EXP produzimos uma extensa avaliação que examina o desempenho de várias técnicas de otimização quando submetidas

a condições variadas de desempenho de rede. Entendemos que a plataforma VR-EXP possui potencial para beneficiar dois públicos-alvo distintos. Sob a perspectiva dos desenvolvedores de técnicas de otimização para vídeos VR, esperamos contribuir com uma abordagem útil para conduzir uma avaliação de desempenho precisa e realista quando do desenvolvimento de novas técnicas. Por sua vez, do ponto de vista da operadora de telefonia móvel, esperamos que VR-EXP seja uma ferramenta valiosa para apoiar investigações que visam entender e prever como as condições de desempenho da rede de dados afetam a qualidade da experiência dos assinantes ao utilizar aplicações de vídeo VR.

APPENDIX B — PAPER PUBLISHED AT IEEE GLOBECOM 2016

- **Title**: *Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE*
- **Conference**: IEEE Global Communications Conference (GLOBECOM 2016)
- **Type**: Main track (full-paper)
- **Qualis**: A1
- **Date**: Dec 04-08, 2016
- **Held at**: Washington, USA

Abstract. Due to the fact that video streaming is the current "killer" application and for competitiveness, telecommunication service providers need to be able to answer a fundamental question: to which extent is the available network infrastructure able to successfully provide users with a satisfactory experience when running video streaming applications? Answering this question is far from trivial because existing techniques are neither scalable nor accurate enough. To address this issue, we propose a model to predict video streaming quality based on the observation of performance indicators of the underlying IP network. To accomplish this objective, the proposed model — created using LTE networks as case study — leverages low network consumption active measurements and machine learning techniques. Obtained results show that the proposed solution produces accurate estimates (average error of less than 10%) while keeping intrusiveness around twenty times lower than traditional techniques.

Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE

Roberto Irajá Tavares da Costa Filho, William Lautenschläger, Nicolas Kagami,
Valter Roesler, Luciano Paschoal Gaspary
Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil
Email: {roberto.costa,wrlautenschlager,nskagami,roesler,paschoal}@inf.ufrgs.br

***Abstract*—Due to the fact that video streaming is the current "killer" application and for competitiveness, telecommunication service providers need to be able to answer a fundamental question: to which extent is the available network infrastructure able to successfully provide users with a satisfactory experience when running video streaming applications? Answering this question is far from trivial because existing techniques are neither scalable nor accurate enough. To address this issue, we propose a model to predict video streaming quality based on the observation of performance indicators of the underlying IP network. To accomplish this objective, the proposed model — created using LTE networks as case study — leverages low network consumption active measurements and machine learning techniques. Obtained results show that the proposed solution produces accurate estimates (average error of less than 10%) while keeping intrusiveness around twenty times lower than traditional techniques.**

I. Introduction

The mobile-broadband penetration rate for 2015 was 86.7% in developed countries, while the corresponding fixed-broadband rate reached 29% of the population according to the ITU [1]. Along with the increase in broadband subscriptions, the dissemination of video applications has hampered the management of the quality of provided services. This adversity has been induced mainly by the fact that video applications tend to demand a greater amount of network resources when compared to other applications. Video traffic is projected to comprise nearly 86% of all Internet traffic before the end of 2016 [2].

In this context, the scientific community as well as the industry agree that maximizing the user's Quality of Experience (QoE) regarding video streaming applications represents a relevant research challenge [2]. An essential aspect in this direction is to systematically determine the quality of the provided video services. To this end, service providers require a solution with low intrusion, scalability, and a reasonably accurate way to measure the quality of service delivered. This task becomes particularly challenging if encompassing cellular networks (our focus in this paper), in which highly intrusive measurement techniques have the potential to negatively affect the quality of provided services.

Msakni and Youssef [3] analyzed some prominent, recently proposed techniques for QoE prediction which don't rely on video transfer. They concluded that none of the considered approaches could be deemed reliable. Essentially, the non-linearity of human opinion compromises accuracy when using network parameters as direct predictors of QoE, since a given configuration can be graded differently in Mean Opinion Score (MOS). An alternative strategy would be to evaluate video quality by objectively grading the user's opinion. However, techniques that allow for such a measurement require the transit and analysis of real video files, entailing a substantial increment in network traffic.

In this paper we propose a prediction model for performance indicators of video streaming services based on indicators from the underlying IP network. The proposed model explores decision trees in order to determine the relation between network QoS indicators[1] and objective indicators representative of the video reproduction quality experienced by the end user, henceforth designated as AppQoS[2]. Additionally, by means of AppQoS processing, the model allows an inference of user QoE. This first iteration was focused on LTE networks as a case study. The results obtained suggest that the proposed model is an adequate approach to facilitate large-scale yet accurate prediction of quality of video services with minor intrusion.

The remainder of this paper is organized as follows. Section 2 describes the related work. Section 3 presents the proposed prediction model. Section 4 details the experiment configuration and model construction aspects. Section 5 reports performance results as well as possible model applications. Section 6 shows our conclusions along with our perspective for future work.

II. Related Work

Traditional mechanisms such as Mean Opinion Score (MOS), Perceptual Evaluation of Video Quality (PEVQ) and Peak Signal-to-Noise Ratio (PSNR) are not scalable when applied to systematically evaluate if a network infrastructure provides the required quality to support video streaming applications. This is especially true when considering the high costs involved with user interviews and video downloads. In an attempt to overcome these limitations, recent investigations proposed to estimate quality using alternative data acquisition procedures, compatible with large-scale scenarios.

[1]The term network QoS indicators refers to performance of IP networks.

[2]The term AppQoS refers to the last objective barrier capable of being measured in the context of the end user.

A first group of investigations is characterized by techniques that estimate QoE indicators using network-based measurements. Due to not taking into account application performance, these techniques do not allow an accurate understanding of the application behavior [4], [5]. This occurs because they are biased by the non-linearity of human opinion. Conversely, a second group of related work tries to establish the relationship between application and QoE, without considering network QoS indicators [6], [7]. The lack of connection among AppQoS, QoE and network performance hinders the service provider's ability to understand the network influence on QoE indicators.

In this context, the work proposed in this paper contributes a step forward in the state of the art since, unlike the related work, it proposes the use of network performance measurements to estimate intermediate indicators between QoS and QoE, which we labeled AppQoS. Additionally, once AppQoS is estimated, it can be used to predict user QoE. Therefore, the QoE indicator allows a fast identification of network points suffering quality degradation, while AppQoS indicators provide a finer-grained view of application behavior, allowing the provider to understand the root cause of degradation. Finally, we claim that the proposed model is scalable since it allows a reliable estimation of application performance at a fraction of the network intrusion necessary to perform measurements using real application traffic.

III. LEAP: Lightweight AppQoS and QoE Predictor

In this paper we propose a Lightweight AppQoS and QoE Predictor (LEAP). It is capable of providing a detailed view of how the network performance affects video streaming applications and, moreover, the corresponding user experience. Figure 1 shows the general model scheme. The model is designed to receive four network performance indicators as input: (*i*) delay, (*ii*) jitter, (*iii*) throughput and (*iv*) packet loss. To capture video playback performance, the model predicts three video playout performance indicators: (*i*) startup time, (*ii*) stall count and (*iii*) total stall length. To estimate each AppQoS indicator, the four network QoS indicators are analysed together. In a second stage, the three AppQoS indicators are used to estimate QoE (using the MOS score).

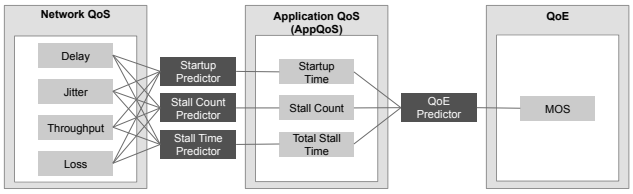


Figure 1. LEAP general scheme

A. AppQoS Prediction

LEAP leverages the Regression Decision Tree technique to construct the three AppQoS prediction models. It was chosen mainly due to its suitability to handle complex and non-linear relationships between attributes. In order to confirm our choice, we conducted experiments considering the following algorithms: Multiple Linear Regression, Artificial Neural Networks, Gaussian Naive Bayes and Support Vector Regression. However, those techniques were outperformed by decision trees when analyzed using the Nested Cross Validation (NCV) protocol [8].

During the training stage, the model learns how each attribute *X* affects the response variable *Y*. In our case, the network QoS indicators are the *X* attributes and the video streaming performance indicators (AppQoS), *Y*. Once the training stage is finished, LEAP is able to estimate AppQoS indicators by comparing the measured QoS indicators against the decision trees. The model evaluates each QoS indicator, node by node, until it reaches a leaf, where the predicted AppQoS is defined. For example, in Figures 2 and 3 it is possible to observe a portion of the decision trees for total stall length and stall count for 1080p videos. Due to space and legibility constraints, we present two partial excerpts from the resulting decision trees. Their constructive aspects and a performance evaluation will be presented in Subsection IV-C.

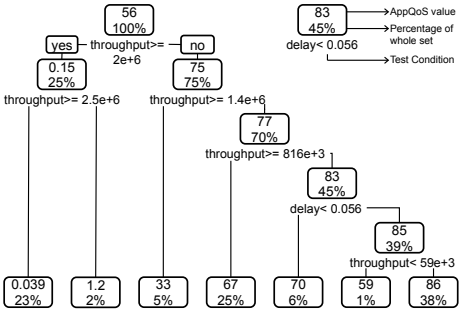


Figure 2. Decision tree for total stall length of 1080p video playout

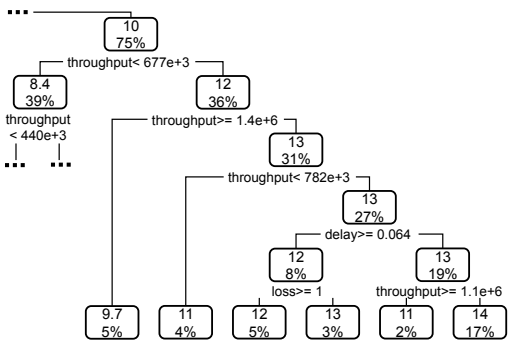


Figure 3. Decision tree for stall count of 1080p video playout

Each decision tree is built using a recursive procedure that executes successive splits, starting from a single node containing all attributes (QoS parameters). This procedure defines the tree growth and is controlled by two parameters. The first is the *minsplit*, which represents the minimum number of related observations for a new branch to be created. The second parameter is called *cp*, which describes the minimum gain, regarding error reduction, that a specific node needs to provide in order to be created. Each resulting decision tree has a particular cross-validation error (*X*), which provides an estimated error when testing the prediction structure against independent data (a portion of the dataset not used during the training stage). Finally, in the pruning stage, each pair of

leaves with a common parent is tested for merge according to the Mean Squared Error (MSE). If MSE is reduced, then those leaves are removed and their parent becomes a leaf node. Otherwise, the structure remains the same for those nodes.

Finally, it is important to emphasize that LEAP is divided into two distinct phases. In the first stage, the tasks with high complexity, such as machine learning technique selection, construction and training of the LEAP model, are performed offline. In particular, the construction of an optimal decision tree is known to be an NP-Complete problem, so a heuristic algorithm is used to obtain a near optimal structure. In turn, the second stage occurs as an online procedure. Whenever QoS indicators are available in the database, a subroutine estimates the AppQoS and QoE indicators just by comparing them against the decision tree thresholds.

B. QoE Prediction

Once the AppQoS indicators are estimated, LEAP can use them to predict a corresponding expected quality of user experience. To establish the relationship between AppQoS and QoE, LEAP employs an adaptation of recent research work, which, based on user interviews, derived a mathematical model for mapping AppQoS to a MOS score. The influence of startup time on MOS is defined by Equation 1, where *Qini* is the resulting MOS score and t_0 is the predicted startup time. The values 0.963 and 5.381 were calculated by solving a nonlinear minimization problem for the mean squared error between MOS scores in t_0 and the function$f(t_0)$ [9], [10].

$$Qini = -0.963 \times \log_{10}(t_0 + 5.381) + 5 \tag{1}$$

Additionally, to allow estimation of the effects of stall count and total stall length on QoE, we first need to define a factor λ. This factor represents the ratio between the video's total stall length σ and the total length of the video playback (the sum of σ and the video duration ρ), as shown in Equation 2. According to Casas *et al.* [11], this observation should be done in time slots T with a typical duration of one minute. This approach allows generalizing the method for videos of any length. Once λ has been calculated, it indicates a degradation level $1 \leq i \leq 5$, according to the λ intervals shown in Table I. Finally, MOS *Qst* is calculated according to Equation 3, where a_i, b_i, c_i are constants defined $\forall i = 1, 2, 3, 4, 5$ according to Table I, and n is the stall count within a specific time slot T. In Equation 3 one can observe that, for a stall count higher than six, the user experience will be fully degraded, resulting in a MOS score 1. We would like to emphasize that the equations presented in this subsection rely on the result of previous research work. The reader interested in their validations may refer to [9], [10], [11].

$$\lambda = \begin{cases} \frac{\sigma}{\sigma+\rho}, & \text{if } \sigma + \rho < T \\ \frac{\sigma}{T}, & \text{otherwise} \end{cases} \tag{2}$$

$$Qst = \begin{cases} 1, & \text{if } n > 6 \\ a_i \times e^{-b_i \times n} + c_i, & \text{if } n \leq 6 \end{cases} \tag{3}$$

Table I
FACTOR VALUES ACCORDING TO λ FOR MOS CALCULATION (*Qst*)

Factor	$\lambda < .05$	$.05 \leq \lambda < .1$	$.1 \leq \lambda < .2$	$.2 \leq \lambda < .5$	$\lambda \geq .5$
a	3.01	3.09	3.19	3.24	3.30
b	0.76	0.99	1.52	1.69	1.88
c	1.99	1.90	1.81	1.75	1.69

IV. MODEL CONSTRUCTION

This section presents the practical aspects related to the construction of the prediction model. First, we describe the acquisition of QoS and AppQoS indicators. Right after, we illustrate details of the training environment. Lastly, we examine the resulting prediction models.

A. Aquisition of QoS and AppQoS Indicators

In order to obtain network performance indicators, we employed an active measurement-based platform named NetMetric [12]. NetMetric works with a "Manager" entity, responsible for configuring schedules to be run in the "Agent" entities. An Agent entity can be used both as the origin of a measurement (source point) or as the target (reflector point). The platform was configured to run groups of two different packet bursts. The first burst makes use of the UDP protocol and unidirectionally measures One Way Delay (OWD), jitter and packet loss by injecting 400 packets of 100 bytes at 50ms intervals. The second burst gauges the throughput using the TCP protocol with 640 packets of 1,488 bytes. Both bursts amount to 992 KB worth of data. Although NetMetric is capable of deriving bidirectional metrics, we have chosen to confine our experiment to unidirectional measurements in correspondence with our case study, the LTE downlink.

For the purpose of obtaining AppQoS indicators, a specific module was developed for NetMetric in the form of a plugin for the Chrome browser. This plugin enabled the extraction of performance indicators related to the reproduction of video through an HTML5 native video player. This choice is justified by the migration of big video content providers to HTML5 technology, such as Youtube and Netflix.

In order to measure within the application layer, a NetMetric source Agent entity periodically reproduced videos via the Chrome browser, which retrieved its files from a reflector Agent. Two video files, of one minute each, were used in the algorithm's training phase. Both were in the MPEG v4 format and coded in H.264. While the first had a 720p resolution and 9.2 MB, the second had a 1080p resolution and 14.3 MB. The transfer of videos of different resolutions is deemed necessary in light of introducing different bitrates, which demand varying degrees of performance from the underlying network. Therefore, each of the three AppQoS predictors needed to be trained separately in order to be acquainted with the network demands of each bitrate.

B. Training Environment

A possible approach for obtaining a dataset relating QoS and AppQoS parameters would be to run the experiment on a deployed LTE network, measuring both the network and application layer indicators simultaneously in an effort to guarantee corresponding conditions in both layers. However, considering the invasive nature of these two measurement techniques, their

concomitant execution would result in mutual interference stemming from a fierce competition for the shared resources. Another possible approach would be to serialize the measurements. Yet, due to a considerable variability in the performance of LTE networks, detectable even in short time intervals, we decided against using such training configuration. The training environment demands stable conditions between the evaluation of the network and application layers. Otherwise, the learning algorithm could establish inaccurate correlations concerning the influence of each network parameters towards the target variables.

In consideration of these difficulties, we opted to devise a controlled environment capable of faithfully emulating the network conditions observed in the deployed setting. In order to reproduce conditions such as throughput, delay, jitter and packet loss present in an LTE environment, we made use of WANEM [13]. The WANEM tool allows us to impose specific constraints on a target network while keeping the network conditions static between consecutive measurements. Figure 4 depicts the topology used to set our controlled environment up.

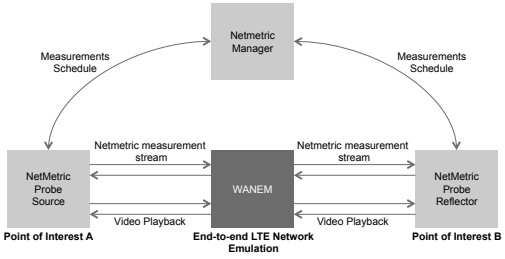


Figure 4. Controlled environment setup

The selection of parameters used in the WANEM configuration was based on 7,450 measurements taken by NetMetric in an LTE network deployed countrywide[3], between May and October of 2015. Table II summarizes the four levels chosen for each of the QoS indicators. The QoS indicators were individually tested as to the normality of their distributions via the Shapiro-Wilk test to a significance level of $\alpha = 0.05$. On account of these normal distributions, the delay, jitter, and throughput indicators have been segmented using quartile analysis. The values associated with the packet loss indicator were selected through the use of modal analysis for integer levels.

A first round of experiments (via *Full Factorial design* $2^{k.r}$) expressed that the jitter indicator did not contribute to the regression model proposed in the previous section. Once the relevant parameters have been determined (throughput, delay, and packet loss), we performed a subsequent set of experiments, this time with *Full Factorial design* $4^{k.r}$, thus allowing a greater degree of variation for each parameter without incurring in an unmanageably large number of experiments. With three input parameters and four levels, the *design* resulted in 64 possible combinations. After observing the variation in the results, we defined the number of ten repetitions for each combination, considering a significance level of 95%.

[3]For confidentiality reasons, we are not allowed to disclose a detailed characterization of the deployed network.

Table II
LTE INDICATORS USED IN THE MODEL TRAINING STAGE

Indicator	Value
Throughput	0.9 Mbps; 8.8 Mbps; 15.5 Mbps; 25.3 Mbps
Delay	22 ms; 56 ms; 64 ms; 98 ms
Loss	0%; 2%; 5%; 13%

C. Resulting Predictors

At this stage, after acquiring data from the aforementioned controlled environment, we focused on generating binary decision trees for each of the three AppQoS indicators. These trees were specifically made for the two different resolutions (720p and 1080p). Considering these two details, we had a total of six tree combinations.

The trees represented in Figures 2 and 3, along with the rest of the trees generated in the scope of this work, display the optimal pruning for these structures. Figure 2 shows the tree for total stall length in 1080p resolution as a function of the three primary network indicators. The root node, which represents the first decision, makes use of the throughput as the deciding criterion (represented in bits per second). The value present in the node, that of 2 Mbps, is associated with the bitrate used in the test videos - around 1.9 Mbps. For higher throughput values, the average stall length (0.15 seconds, according to the value indicated by the root left child node) is considerably less than the observed for networks with a lower throughput (75 seconds, as expressed by the node to the right of the root). This observation implies that a network being capable of supplying the video bitrate is the most important factor in determining the overall video behavior.

Another related aspect is the existence of an intermediary range above the bitrate, in which there is a slight degradation for video playback performance indicators - in the case of Figure 2, between 2 Mbps and 2.5 Mbps. In this range, instantaneous throughput variations may lead to an empty buffer interruption, which leads to degradation. For values above 2.5 Mbps, however, the video reproduction develops mostly without interruptions, independent of any other predictive indicator. It should be noted that a similar behavior was observed with the 720p resolution, in which the root node was determined by a throughput threshold identical to the video bitrate of 1.2 Mbps, even though a flawless reproduction was only identified for throughput values above 1.9 Mbps.

The root node of the 1080p stall count indicator tree along with the left portion of the tree (not shown in Figure 3) are analogous to their aforementioned total stall length counterparts. This means the first decision is taken concerning the throughput in relation to the bitrate, and that higher throughput values presented degradation up to a second threshold (2.3 Mbps). However, the branch shown in Figure 3, which represents the portion of the tree immediately connected to the right of the root node, presented a distinct behavior. For this subtree, throughput values between 677 Kbps and 2 Mbps (limited by the root node) predicted an average of 12 stalls, a greater amount than observed for smaller throughputs, which averaged 8.4 stalls. An analysis taking total stall length into consideration shows that, for samples with a throughput below

677 kbps, the videos are expected to stay halted for an average of 83.76 seconds. The remainder of cases presented an average of 64.98 seconds. Thus, we can conclude that the video halts for effectively longer in networks with reduced throughput, even though the stall count is smaller, representing longer-lasting individual interruptions.

V. Performance Evaluation

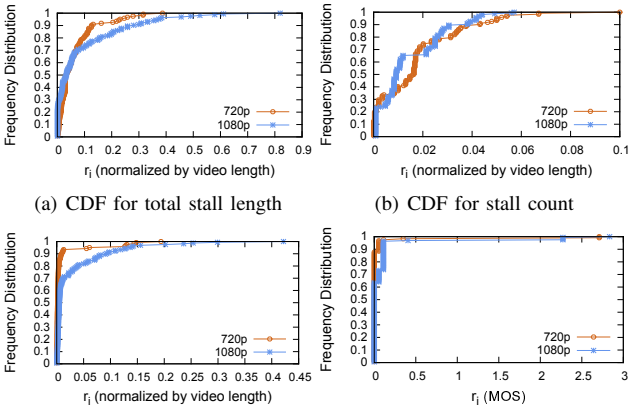
This section presents a performance evaluation of the proposed model. Additionally, this section addresses the results gathered from applying the model to the mobile network mentioned in the Subsection IV-B. Essentially, our objective is to answer the following questions: (*i*) How accurately can the model predict video application behavior based on network performance observations? (*ii*) To which extent is network intrusion kept low when the model is applied to a large scale LTE network? (*iii*) How can providers capitalize on prediction of video quality and QoE?

A. Model Accuracy

With the purpose of determining accuracy, the proposed model was subjected to a test dataset, independent of the training dataset. Each test sample i contains a measurement for each of the three predictor variables (throughput, delay and loss) and three application indicators (stall length, stall count and startup time). By applying the first group of variables as input for the decision trees represented in Figures 2 and 3, we obtained an estimation for each of the variables in the second group, which then have an associated observed value x_i and a predicted value $\hat{x}_i$. This allowed us to calculate, for each sample i, the normalized residuals r_i, defined by the equation $r_i = |\hat{x}_i - x_i|/N$. We use the factor N to normalize the error values. For each of the three application indicators, the value of N is derived from the duration of the videos employed in the controlled environment (60 seconds in this case), which enables a generalization of the evaluation method for videos of any duration.

Figures 5(a), 5(b), 5(c) and 5(d) depict the variation of r_i in the horizontal axis, associating each value in this axis to a portion of the sampling group (in the vertical axis) in which r_i is lesser or equal to the set threshold. Thus, considering the 1080p resolution in Figure 5(c) (video startup time), the value of 0.093 in the horizontal axis is related to the 0.9 value in the vertical axis, indicating that 90% of the samples of the corresponding group have $r_i \leq 0.093$. In the same figure, 90% of the samples with 720p resolution have $r_i \leq 0.0086$. This can be interpreted as an error of 9.3% and 0.86%, respectively. The average of r_i in 90% of the samples for all indicators, considering both 1080p and 720p, has a value of $\bar{r_i} = 0.0982$ (9.8%). We deem this error rate to be satisfactory and in line with recent research work.

Based on the predicted and observed values for the application indicators, it is possible to calculate a MOS value according to the approach presented in Subsection III-B. For this indicator (MOS), the following error rates are depicted in an absolute scale, which ranges from 1 to 5. According to Figure 5(d), MOS prediction presented an error rate of up to $r_i = 0.11$ for 90% of samples for videos of 1080p. This means that, for example, if a $MOS = 3.5$ is calculated using the observed application indicators, a value of $3.39 \leq \widehat{MOS} \leq 3.61$ is to be predicted 90% of the time. The error rate was smaller in 720p resolution, with a value of $r_i = 0.05$. Even though the AppQoS estimation error rate is low, this disturbance did not significantly impact QoE prediction, which presented an even lower prediction error, as shown in Figure 5(d).



(a) CDF for total stall length (b) CDF for stall count

(c) CDF for startup time (d) CDF for MOS

Figure 5. AppQoS and QoE prediction error for 1080p video playout

B. Model Intrusiveness

Figure 6 illustrates a comparison of intrusiveness, constructed with traffic volume parameters described in Subsection IV-A, between the proposed model and measurements taken through use of real video transfers. It should be noted that the most intrusive LEAP case (500 probes with a polling interval of 10 minutes) generates 66.65 GB/day. With an equivalent configuration, the real video transfer strategy would require 1,574.71 GB/day. Therefore, the adoption of LEAP is on average one order of magnitude less intrusive.

Complementing the aforementioned analysis, take into account that the unstable performance of LTE networks, mainly characterized by user mobility and fierce competition of radio interface resources, demands a reduced monitoring polling interval to achieve a realistic view of the network environment. In this setting, low intrusion is an essential requirement to be met. Otherwise, the generated traffic would consume an enormous amount of resources, undermining the availability of resources for end users.

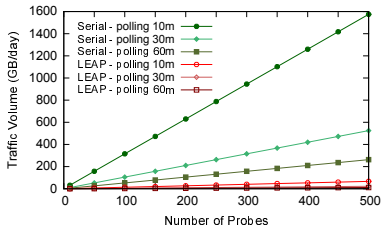


Figure 6. LEAP scalability

C. LEAP in Action

After its creation, the model was deployed in one of the largest mobile operators in Brazil, already mentioned in

Subsection IV-B. We installed reflector agents as close as possible to the mobile operator's main CDN nodes responsible for delivering cached videos. With this setup, the model was capable of measuring AppQoS and QoE in an end-to-end approach, from an application server down to the end user premises. In order to decide where to install the source agents, we examined the operator's traffic matrix and concluded that a Pareto's Principle fit was reasonable, that is, 80% of all mobile data traffic was generated in less than 20% of eNode Bs (a.k.a. Base Station). Then, the source agents were deployed close to each selected eNB in decreasing order of traffic volume.

Due to space constraints, we show just a single real world application of LEAP. Using the parallel coordinates visualization technique, Figure 7 shows a LEAP graph that illustrates the relationship between QoS, AppQoS and QoE, for a particular source agent. In this figure it is possible to observe that both curves share a QoS condition: 77 ms delay, 0% loss and 1.950 Mbps throughput. However, when LEAP predicts the AppQoS performance and QoE, we verify a very divergent performance between resolutions. For 720p, the predictor estimates zero stall count and a startup time of 290ms, incurring in a MOS of almost 5. However, when predicting 1080p performance for the same network, LEAP predicts a startup time of 910ms, 9.68 stalls with a total length of 32.9 seconds, which implies an MOS score of 1.75. In this particular case (and in other interest points), LEAP provided an integrated, real-time view of what the network is delivering and the corresponding effects on video playout performance, *i.e.*, expected user experience. Moreover, through analysis of the respective decision tree, LEAP was able to provide the notion of how QoS parameters need to be tuned in order to achieve the desired service quality.

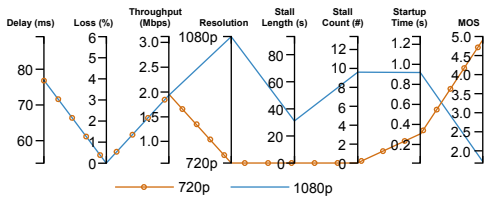


Figure 7. Integrated QoS, AppQoS and QoE view

VI. Conclusion and Future Work

The results obtained suggest that it is feasible to estimate AppQoS and QoE for video streaming applications using QoS indicators as predictors. The estimated parameters achieved an average error below 9.92%, and a MOS estimation error below 11% for over 90% of cases. Furthermore, LEAP requires less than 4% of the traffic volume when compared to traditional techniques. The low intrusiveness allows the service provider to configure systematic measurements, with reduced polling interval, without an excessive usage of network resources.

To maintain predictor accuracy over time, LEAP needs to be periodically recalibrated. This event is triggered by a scheduled procedure named Accuracy Check Probe. This procedure executes a strategically distributed set of reference measurements using the LEAP Browser Plugin and compares the measured AppQoS values against the predicted values. This routine assesses the prediction error and, whenever necessary, triggers the model recalibration procedure.

The LEAP model was shown to be an efficient tool to aid network operators in tuning and troubleshooting routines. LEAP provides an indication of which video resolution will fit into a given network segment. For this reason, and to keep focus on network performance, we decided to operate under fixed bitrates. As future work, we intend to evolve the model to provide a complementary prediction considering adaptive video streaming mechanisms like DASH. Additionally, we intend to assess LEAP's generality by applying it to other applications such as VoIP and online games. Finally, we plan to specify a methodology and conduct extensive interview-based tests aiming at advancing the state of the art towards more accurate mapping strategies of AppQos to QoE.

Acknowledgement

This work has been supported by CNPq (project 482008/2013-0) and PROPESP - IFSul.

References

[1] ITU, "Measuring the information society," 2015. [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2015/MISR2015-w5.pdf.

[2] M. Katsarakis, G. Fortetsanakis, P. Charonyktakis, A. Kostopoulos, and M. Papadopouli, "On user-centric tools for qoe-based recommendation and real-time analysis of large-scale markets," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 37–43, 2014.

[3] H. Msakni and H. Youssef, "Is qoe estimation based on qos parameters sufficient for video quality assessment?" in *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, July 2013, pp. 538–544.

[4] T. De Pessemier, K. De Moor, W. Joseph, L. De Marez, and L. Martens, "Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching," *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 47–61, 2013.

[5] Z. Xiao, Y. Xu, H. Feng, T. Yang, B. Hu, and Y. Zhou, "Modeling streaming qoe in wireless networks with large-scale measurement of user behavior," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[6] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an internet video quality-of-experience metric," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, 2012, pp. 97–102.

[7] ——, "Developing a predictive model of quality of experience for internet video," in *Proceedings of the ACM SIGCOMM 2013*, 2013, pp. 339–350.

[8] I. Tsamardinos, A. Rakhshani, and V. Lagani, "Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization," in *Artificial Intelligence: Methods and Applications*, 2014, pp. 1–14.

[9] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2014.

[10] 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 *Proceedings of the 4th International Workshop on Quality of Multimedia Experience (QoMEX)*, 2012, pp. 1–6.

[11] P. Casas, R. Schatz, and T. Hoßfeld, "Monitoring youtube qoe: Is your mobile network delivering the right experience to your customers?" in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 1609–1614.

[12] G. L. dos Santos, V. T. Guimaraes, J. G. Silveira, A. T. Vieira, J. A. de Oliveira Neto, R. da Costa, and R. Balbinot, "Uama: a unified architecture for active measurements in ip networks; end-to-end objetive quality indicators," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2007, pp. 246–253.

[13] H. K. Kalitay and M. K. Nambiar, "Designing wanem: A wide area network emulator tool," in *Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS)*, 2011, pp. 1–4.

APPENDIX C — PAPER PUBLISHED AT IEEE INFOCOM 2018

- **Title**: *Scalable QoE-aware Path Selection in SDN-based Mobile Networks*
- **Conference**: IEEE International Conference on Computer Communications (INFOCOM 2018)
- **Type**: Main track (full-paper)
- **Qualis**: A1
- **Date**: Apr 15-19, 2018
- **Held at**: Honolulu, USA
- **Notes**: Best-in-Session Presentation Award

Abstract. To deal with the massive traffic produced by video applications, mobile operators rely on offloading technologies such as Small Cells, Content Delivery Networks and, shortly, Cloud Edge and 5G Device to Device communications. Although these techniques are fundamental for improving network efficiency, they produce a multitude of paths onto which the user traffic can be forwarded. Thus, a critical problem arises about how to handle the increasing video traffic while managing the interplay between infrastructure optimization and the user's Quality of Experience (QoE). Solving this problem is remarkably difficult, and recent investigations do not consider the large-scale context of mobile operator networks. To address this issue, we present a novel QoE-aware path deployment scheme for large-scale SDN-based mobile networks. The scheme relies on both a polynomial-time algorithm for composing multiple QoS metrics and a scalable QoS to QoE translation strategy. Considering real mobile operator network and video traffic traces, we show that the proposed algorithm outperformed state-of-the-art approaches by reducing impaired videos in aggregate MOS by at least 37% and lowering accumulated video stall length four times.

Scalable QoE-aware Path Selection in SDN-based Mobile Networks

Roberto Irajá Tavares da Costa Filho, William Lautenschläger, Nicolas Kagami,
Marcelo Caggiani Luizelli, Valter Roesler, Luciano Paschoal Gaspary
Federal University of Rio Grande do Sul (UFRGS), Brazil
{roberto.costa,wrlautenschlager,nskagami,mcluizelli,roesler,paschoal}@inf.ufrgs.br

***Abstract*—To deal with the massive traffic produced by video applications, mobile operators rely on offloading technologies such as Small Cells, Content Delivery Networks and, shortly, Cloud Edge and 5G Device to Device communications. Although these techniques are fundamental for improving network efficiency, they produce a multitude of paths onto which the user traffic can be forwarded. Thus, a critical problem arises about how to handle the increasing video traffic while managing the interplay between infrastructure optimization and the user's Quality of Experience (QoE). Solving this problem is remarkably difficult, and recent investigations do not consider the large-scale context of mobile operator networks. To address this issue, we present a novel QoE-aware path deployment scheme for large-scale SDN-based mobile networks. The scheme relies on both a polynomial-time algorithm for composing multiple QoS metrics and a scalable QoS to QoE translation strategy. Considering real mobile operator network and video traffic traces, we show that the proposed algorithm outperformed state-of-the-art approaches by reducing impaired videos in aggregate MOS by at least 37% and lowering accumulated video stall length four times.**

I. Introduction

Context and Motivation. The future is mobile: wireless networks will account for two-thirds of all IP traffic by 2020 [1]. In this context, operators are being challenged by video traffic, which is pushing their network infrastructure to the limit [2]. According to Cisco, mobile video accounted for 60% of total Internet traffic in 2016. And there is more to come: since mobile video is expected to increase 9-fold by 2021, reaching 78% of total data traffic [3]. One key enabler to support such amount of video traffic is the improvement in connection speed. The average downlink speed grew more than 3-fold, from 2.0 Mbps in 2015 to 8.3 Mbps in 2016 [1], allowing the network to successfully deliver high quality videos (*e.g.*, 720p and 1080p).

To deal with the massive growth in data traffic, mobile operators have to constantly invest (*i.e.*, CAPEX and OPEX) to increase capacity, to switch technology (*e.g.*, 3G, 4G, 4G+, 5G), as well as to improve outdoor and indoor coverage. In the opposite direction, the Average Revenue Per User (ARPU) for mobile broadband has fallen from USD 23 in 2013 to USD 13 in 2015 [1]. All those elements together place a lot of pressure on operators to manage their infrastructure efficiently [2].

Aiming at increasing efficiency, mobile operators have been relying on offloading technologies such as Small Cells (Femtocell, Picocell, Microcell), Wi-Fi offloading, Content Delivery Networks (CDNs), and, in the near future, 5G Device-to-Device communication (D2D) and 5G Mobile Edge Computing [2], [4], [5]. As shown in Fig. 1, these technologies are capable of offloading different segments of the network (*i.e.*, edge, aggregation, core and peering) and play a fundamental role in network infrastructure optimization, as they shorten the distance between subscriber and content while avoiding network congestion by spreading the traffic among alternative paths. As an indication of how important these offloading technologies are, in 2016 60% of mobile data was relocated to alternative paths, just considering Wi-Fi and Femtocell offloading [3].

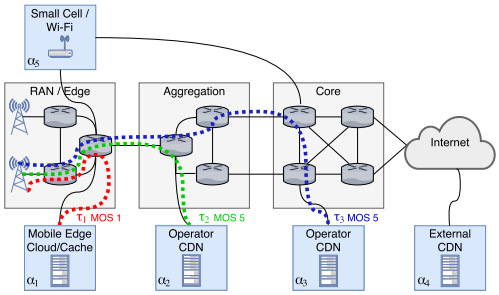


Fig. 1. Mobile operator network and the path diversity increased by the offloading techniques

Problem. The adoption of offloading techniques introduces a multitude of possible paths onto which user traffic can be forwarded and, as an immediate consequence, raises the complexity of the network management (*e.g.*, path selection, configuration, troubleshooting). While very important, such an advanced infrastructure does not directly translate into improved Quality of Experience (QoE) [6]. This is notably true if considering that some offloading techniques may rely on shared and/or third-party infrastructure, which would possibly exacerbate the unpredictability regarding the delivered quality of experience. Given the above, the fundamental problem lies in how to handle the increasing end-user traffic while optimizing infrastructure utilization and managing user's quality of experience. In fact, from the operator's perspective, solving this problem is crucial for improving competitiveness, since the effective management of the interplay between perceived QoE and infrastructure investments is the main factor for increasing return of investment [7], [8]. In a recent investigation [6], the authors proposed an approach for addressing

the QoE-aware path selection problem at the content provider premises. They proposed an inter-AS path selection scheme based on performance indicators measured at the *server side* (*i.e.*, Facebook server logs) built on top of an SDN-based infrastructure. Aiming at addressing the QoE-aware problem at the *operator side*, in this paper we propose SQAPE - **S**calable **Q**oE-**a**ware **P**ath **Se**lection scheme to be used in large-scale SDN-based mobile networks.

Research Challenges. To illustrate the aforementioned QoE-aware path selection problem, let us consider the simplified network topology shown in Fig. 1. In this example, which is a temporal snapshot of a dynamic scenario, a subscriber requests a specific video (1080p and 4 Mbps of bitrate) that can be served by any of the three offloading containers (namely α_1, α_2 and α_3). Consider the paths τ_1, τ_2 and τ_3 have the following characteristics $\Phi(\tau) = \{$hop count, residual bandwidth[1] (Mbps), delay (ms), loss[2] (%)$\}$: $\Phi(\tau_1) = (3, 3, 5, 0)$, $\Phi(\tau_2) = (4, 10, 10, 0)$ and $\Phi(\tau_3) = (7, 100, 15, 0)$. In light of this scenario, mobile operators have to address the following two questions: (*i*) how to accurately predict video QoE for given pairs of source and destination in large-scale networks? (*ii*) How to use such a QoE indicator to dynamically select and deploy QoE-aware paths that minimize infrastructure utilization over time? Despite some very important investigations into this field (please refer to Section V for details), these two questions remain unanswered and are briefly discussed next.

QoE Estimation. In order to answer Question (*i*) it is crucial to define how to estimate QoE for a video if delivered through a given network path. Several research investigations proposed strategies based on QoE estimation for path selection. However, those approaches are not suitable for mobile operators because they rely on QoE information which is not known by the operator (*e.g.*, server-side or client-side indicators). Additionally, even if those indicators were available, such information only allows predicting end-to-end QoE, which is not appropriated for supporting path selection decisions, since it does not provide indications regarding impaired links.

The first group of studies makes use of QoE information derived from end-user devices [6], [9], [10], [11], which entails lower system scalability, reduced flexibility and the requirement of end-user collaboration through the installation of measurement software on their devices [12]. A second group relies on server-side information [13], [14], which is usually owned by the content provider and is not available for the operator even when the CDN node is deployed on the operator premises. Finally, a third group employs observation of video streaming data traffic inside the network [15], [16]. Yet, considering the widespread adoption of HTTPS for video streaming delivery, approaches based on video streaming flow analysis inside the network become unpractical [17].

Indeed, there is no fast track to estimate QoE for video streaming service. One promising strategy makes use of in-network QoS measurements to predict QoE, since most network providers already have tools to measure those indicators. However, those techniques need additional effort as they still represent a challenge from the provider's perspective. [18].

Path Selection. With regard to Question (*ii*) – namely how to coordinate QoE-aware path selection and resource utilization minimization – let us consider the adoption of the end-to-end QoS to QoE mapping function we proposed in a previous work [19]. In this case, one would use residual bandwidth, delay and packet loss, so it would be possible to estimate MOS for the paths τ_1, τ_2 and τ_3 shown in Fig. 1. In this case, differently from τ_2 and τ_3, τ_1 does not provide the QoS performance needed to achieve MOS 5. Also, τ_2 should be preferred since τ_3 does not improve QoE and uses more resources. However, when considering real-world networks, such an end-to-end approach leads to the evaluation of an immense number of paths, which is impractical. One possible approach to solve this issue would be to consider paths as a composition of N links while using MOS estimation as link weight. However, the QoS metrics used to estimate QoE have different composition rules, which lead to an inconclusive aggregation result. In other words, the composition of N links with MOS 5 does not necessarily result in an MOS 5 path [20].

Another alternative would be to compose the multiple QoS metrics along the path and then apply the MOS estimation when the path is finally composed. However, the composition of one concave, one additive and one multiplicative metric was shown to be NP-Complete [20].

Finally, it would be possible to employ one of the following strategies: (*i*) use a single QoS metric composition (*e.g.*, residual bandwidth), since it is the most influential when predicting QoE [17], or (*ii*) use delay, which is an excellent metric for sensing queue occupation [20] or, finally, (*iii*) a combination of both these metrics used in conjunction with a modified version of Dijkstra, which has shown to be a special case of aforementioned NP-Complete problem, which has polynomial solution, named Shortest-Widest Path (SWP) [20]. Considering that those approaches are effective in avoiding bottlenecks, they consequently improve video QoE. However, from a mobile operator's perspective, they fail at a critical point: they perform optimal decisions only within a time frame and do not take the long-term operation or the infrastructure resource utilization into consideration.

Proposed Approach and Contributions. In this paper we propose a novel scheme for network path provisioning that employs active QoS measurements to predict video streaming performance and QoE, which will be used for deployment of QoE-aware paths in an SDN-enabled[3] mobile network. The work proposed in this paper contributes significantly to the state of the art since, unlike the related work, it does not rely on third-party information. Our approach considers neither video streaming flow data nor client/server-side information. The proposal is able to provide QoE-aware paths, in networks

[1]In this paper we consider residual bandwidth, TCP throughput and Bulk Transfer Capacity (BTC) as equivalent.

[2]In this paper we consider delay and packet loss as one-way metrics.

[3]Although this work can be adapted to other underlying technologies, such as BGP and MPLS-TE, the OpenFlow protocol was chosen due to its ability to orchestrate network resources in a centralized manner, with a complete view of the network state, and its abstractions to handle packet forwarding.

characterized by high path diversity, with high accuracy and low network resource consumption. The contributions of this work are summarized as follows.

- **MOS-based link composition**. We propose an innovative approach to combine QoS measurements and MOS estimation in a link composition scheme. The proposed strategy allows predicting MOS in end-to-end paths by composing multiple QoS metrics.
- **Large-scale path selection algorithm**. We present a heuristic algorithm capable of finding optimized QoE-aware paths in polynomial time. The proposed algorithm runs on top of an SDN architecture and takes advantage of the northbound interface to coordinate related mechanisms such as QoS active measurements, QoE prediction and network rule management.
- **Realistic evaluation**. We present an evaluation considering real mobile operator topology and video traffic traces, where the proposed heuristic algorithm was found to outperform state-of-the-art approaches.

The remainder of this paper is organized as follows. In Section II we present the QoE-aware path selection problem. In Section III we present SQAPE as well as design aspects regarding path selection in large-scale networks. In Section IV we present the performance evaluation using realistic topology and workload. Section V presents an overview of relevant research work with respect to QoE-aware path selection. We conclude this paper summarizing our findings in Section VI.

II. Problem Formulation

We start by formally defining the QoE-aware path selection problem. Note that, at this point, we aim to estimate QoE according to a QoS to QoE mapping function. Then, we combine it with flexible SDN-based routing mechanisms in order to maximize the overall MOS. Next, we define the inputs and outputs related to our model.

The physical network infrastructure is represented by a direct graph $G = (N, L)$, where N is the set of nodes (*i.e.*, SDN-enabled forwarding devices) and L is the set of links, such that $L \subseteq (N \times N)$. Links are asymmetric and, therefore, bidirectional links are represented as a pair of arcs ((i, j) and (j, i)). We denote the size of sets N and L by $|N|$ and $|L|$, respectively. Each link $(i, j) \in L$ is associated with QoS measurements (*i.e.*, delay, TCP throughput and packet loss). We assume QoS-related data is gathered from the network infrastructure periodically so as to ensure the accuracy and freshness of our model. Therefore, to each link $(i, j) \in L$ is associated a set of functions defined as follows. Function $D : (N \times N) \rightarrow \mathbb{R}_+$ is used to denote measured delay associated with a given physical link $(i, j) \in L$. In turn, function $T : (N \times N) \rightarrow \mathbb{R}_+$ associates link (i, j) with its maximum measured TCP throughput. Last, function $L : (N \times N) \rightarrow [0, 1]$ associates observed packet loss between nodes i and j.

C defines a set of available video content. An element $c \in C$ represents, for instance, a specific video content. For each content c, a set of offloading locations are known in advance. We represent the set of offloading locations for a given content c as the set P and, therefore, $P(c) \subseteq N$.

A path between two distinct nodes i and j consists of a finite sequence of nodes $\tau = \{n_0, n_1, \cdots, n_h\}$ such that $(n_i, n_{i+1}) \in L (0 \leq i \leq h - 1)$. A path τ is simple if all of its nodes are distinct. We denote a valid path between i and j as $\tau(i, j)$ and its corresponding length by $|\tau|$. The set of all simple paths between i and j is denoted by $\Psi(i, j)$. For ease of presentation, end-to-end QoS measurements of a given path τ are associated similarly to previous definitions. Then, functions $T_p(\tau), D_p(\tau)$, and $L_p(\tau)$ represent, respectively, TCP throughput, delay and packet loss. We derive these QoS indicators similarly to Wang and Crowcroft [20]. The end-to-end path TCP throughput (*i.e.*, $T_p(\tau)$) consists of the minimum TCP throughput observed over all links in τ. Formally, it is defined as:

$$T_p(\tau) = \mathbf{Min}_{(0 \leq i \leq |\tau|-1)} T\big(\tau[i], \tau[i+1]\big) \tag{1}$$

Next, the end-to-end path delay (*i.e.*, $D_p(\tau)$) comprehends the additive sum of measured delays (on links) along the path τ. Therefore, the end-to-end path delay is formalized as:

$$D_p(\tau) = \sum_{(0 \leq i \leq |\tau|-1)} D\big(\tau[i], \tau[i+1]\big) \tag{2}$$

Lastly, the end-to-end packet loss estimation (*i.e.*, $L_p(\tau)$) follows a similar strategy to the one proposed by Dobrijevic *et al.* [21]. In their work, the authors estimate end-to-end packet loss based on observed losses in forwarding devices. Here, for higher accuracy, we adopt a strategy that is based on packet loss observed on links (*i.e.*, including forwarding devices and the physical medium). Similarly to Dobrijevic *et al.*[21], we multiply observed packet loss along the links used on path τ (Equation 3).

$$L_p(\tau) = 1 - \prod_{(0 \leq i \leq |\tau|-1)} 1 - L\big((\tau[i], \tau[i+1]\big) \tag{3}$$

Given the above definitions of QoS path composition, we consider our previous work [19] to infer the MOS value for a given path τ. We consider function $\Phi(T_p(\tau), D_p(\tau), L_p(\tau)) \in \mathbb{N}^+$, which correlates end-to-end QoS indicators of a single path τ with the predicted MOS. The Φ two-step function employs decision trees to predict video streaming application performance based on observed network QoS. In a second step, the same function provides a QoE estimation based on the predicted performance for the application layer. The interested reader may refer to [19] for additional information.

Our model considers a set of multimedia connection requests S at a specific time frame t. A multimedia connection request $(i, c) \in S$ represents that a device attached to the infrastructure node $i \in N$ is requesting video content $c \in C$. Notice that content $c \in C$ is potentially available at multiple offloading locations – defined by set $P(c)$. Then, $\forall (i, c) \in S, \forall j \in P(c)$: $\exists (i, j) \in (N \times N)$.

Now we can formally define our problem. Given the inputs

of the model, that is, the infrastructure G with its associated metrics and a set S of multimedia connection requests, the problem consists of finding a valid simple path for each $(i, c) \in S$ in given time slot t so as to *maximize* observed MOS (Equation 4). Therefore, the output of the model is a set of paths that maximize the overall observed MOS given a set of multimedia requests S.

$$\textbf{Max.} \sum_{\forall (k,l) \in S} \sum_{j \in P(l)} \sum_{\forall \tau \in \Psi(k,j)} \Phi\Big(T_p(\tau), D_p(\tau), L_p(\tau)\Big) \quad (4)$$

The model presented in this subsection is used as an important building block of our proposed approach (Section III). SQAPE provides mechanisms to actively monitor the network infrastructure and keeps updated QoS link measurements that are used as input in our model.

III. SQAPE

In this section, we present the main aspects of SQAPE. We begin by introducing the SQAPE architecture. Right after, we describe the QoS measurement strategy and the method for QoE estimation. Finally, we present the SQAPE main algorithm and highlight its most essential characteristics.

A. SQAPE Architecture

SQAPE takes advantage of the centralized control strategy of the SDN/OpenFlow architecture to provide fine-grained control for per-user QoE-aware path deployments across the network. SQAPE architecture consists of three main components: Decision, QoS Measure and QoE Predictor. These components are decoupled from the SDN controller, which allows integration of any controller instance through the northbound interface (NBI). We claim the proposed scheme to be flexible since it employs loosely coupled microservice-based components and can be orchestrated orthogonally to SDN deployments. For example, both the QoS Measure and the QoE Predictor components are coordinated by the Decision one. Considering the above and in consonance with recent investigations [22], [23], SQAPE has the potential to cope with incremental SDN deployment strategies.

For didactic reasons, in Fig. 2 we show the start-up process, which is a special case where SQAPE components interact in a sequential order. After the start-up, events 1-4 run on a periodic basis according to the QoS monitoring frequency (*e.g.*, 60 seconds) and are responsible for determining QoS performance by means of active measurements (Fig. 2 (a)). In parallel, events 5-8 will occur according to the incoming video requests (Fig. 2 (b)). At this stage, SQAPE relies on a QoS composition approach followed by a QoS-to-QoE mapping function to compute per-path MOS estimation. This MOS estimation, along with a link utilization heuristic, composes the distance metric which, ultimately, determines the path to be deployed.

B. From QoS Measurement to QoE Prediction

In this work we consider the use of active measurement methods to assess residual bandwidth, delay and loss. These metrics were selected as the ones most influential to video Quality of Experience according to previous investigations regarding QoS to QoE mapping functions [19], [24], [25]. Active measurement-based methods are particularly interesting as most mobile operators already have tools in place for measuring network QoS indicators.

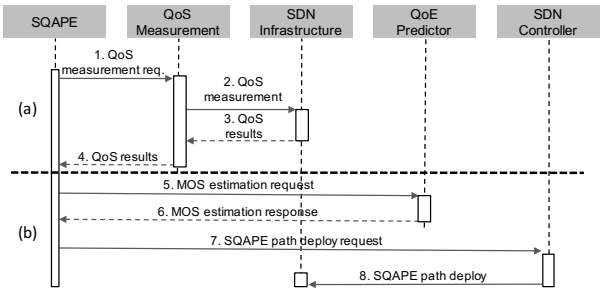


Fig. 2. SQAPE sequence diagram

As discussed in the Introduction and formalized in Section II, we perform measurements in the scope of each link and later compose them to form an estimation of the complete path. Using the topology shown in Fig. 3 as an example, this design choice requires just 94 one-way measurements (*i.e.*, twice the number of links) as opposed to the 22,536 one-way measurements that would be required if we were to measure every possible path between source and destination. As one can observe, the huge number of measurement operations makes path-based approaches impractical for supporting routing decisions in large-scale networks. Also, link-based composition combined with active measurements entails lower network overhead, which is a crucial requirement for scalable monitoring. In practice, per-link measurements require the existence of measurement agents at each vertex. They may be separated by more than a physical link and are represented by the SDN switches, where routing decisions are enforced.

C. SQAPE Algorithm

The SQAPE algorithm focuses on finding a feasible set of paths with maximum MOS over time. For that, SQAPE attempts to determine a suitable balance between MOS and resource consumption in the infrastructure. One component of SQAPE focuses on MOS maximization, which is a specialization of the *Widest Path problem* [20]. Another component targets resource consumption, approaching the well-known minimum path problem. Therefore, SQAPE is a hybrid combination of these two algorithms as it considers both metrics simultaneously.

Algorithm 1 consolidates the proposed path selection approach. As one can observe, the measurement phase collects the results of the latest measurement snapshot and updates the QoS graph for each of its links (line 2). The path selection algorithm operates on top of an adaptation of the Dijkstra shortest-path algorithm in which the distance metric is a combination of MOS estimation and a link utilization contribution. The algorithm starts navigating from node k and determines

Algorithm 1 SQAPE - QoE-aware path selection

Input: $G = (N, L)$: network infrastructure
Input: $(k, l) \in S$: video content request
Input: $P(l) \subseteq N$: set of available network nodes (caches) with content l
Input: α: path selection factor; β: throughput estimation factor
Input: M_{max}: maximum MOS value
1: **for** every link $(i, j) \in L$ **do**
2: *gather* measured QoS indicator for $T(i, j)$, $D(i, j)$, $L(i, j)$
3: **for** every node $u \in N$ **do**
4: $\chi_{[u]} \leftarrow \infty$
5: $\varphi_{[u]} \leftarrow$ NILL
6: $\chi_{[k]} \leftarrow 0$
7: **while** $N \neq \emptyset$ **do**
8: $u \leftarrow$ Extract-Min(N, χ)
9: **for** each link (u, v) **do**
10: $\{T_\tau, D_\tau, L_\tau\} \leftarrow$ QoS metrics of path $\tau(s, v)$ passing through u according to equations (1), (2), and (3)
11: $M \leftarrow \Phi\Big(T_p(\tau), D_p(\tau), L_p(\tau)\Big)$
12: $\chi' \leftarrow M + \alpha \cdot \Big(\big(\sum_{\forall (i,j) \in \tau_{(k,u)}} \frac{1}{T(i,j)} \big) + \frac{1}{T(u,v)} \Big)$
13: $\chi' \leftarrow M_{max} - \chi'$ *iff* $\chi' \leq M_{max}$. Otherwise $\chi' \leftarrow 0$
14: **if** $\chi_{[v]} > \chi'_{[u]}$ **then**
15: $\chi_{[v]} \leftarrow \chi'$; $\varphi_{[v]} \leftarrow u$
16: *deploy* path for req. (k, l) based on $\{\chi, \varphi\}$ such that $\chi_{[j]} : j \in P(l)$ is minimum
17: QoS update: $\forall (i, j) \in \{\chi, \varphi\}$: *estimate* $T(i, j)$ according to β
18: **return** χ, φ

the path with the best local distance metric to each node $u \in N$ (lines 7-14). For each neighbour v, we estimate MOS based on QoS measurement to path τ (line 10-11), according to Function Φ detailed in Section II, and then combine it with an estimation of residual bandwidth (line 12). The objective of this strategy is to capture the interplay among: *(i)* path MOS; *(ii)* path length (hop count); and *(iii)* residual bandwidth. Our algorithm simultaneously maximizes MOS and residual bandwidth while minimizing path length. We achieve this by means of linear combination (line 12). Parameter α determines the importance of the residual bandwidth in relation to MOS. At each iteration, the function Extract-Min removes node u from N according to set χ (line 8). Observe that set χ maintains the best solution found so far, while set φ keeps track of nodes belonging to path $\tau(k, l)$ (line 15). After navigating through all nodes $u \in N$, SQAPE deploys a path for the video request (k, l) such that the $\chi_{[j]}$ is minimum over all $j \in P(l)$. In other words, it takes the minimum path between k and an offloading node j in $P(l)$. Then, we deploy the appropriate SDN forwarding rules (line 16).

Concerning the eventual discrepancy between the frequency of measurements and the frequency of routing requests, the accuracy of path selection can be impaired. The impact of newly routed connections on a path would only be acknowledged after the next measurement iteration, whose frequency may depend on network characteristics. In light of this, after a path is selected, each of its links' residual bandwidth value is updated to reflect the impact of adding a load comparable to a TCP connection with a given video bitrate (*e.g.*, 1080p, 720p). Considering the video bitrate is determined by the client implementation, our algorithm relies on a parameter β. In this step, the link's residual bandwidth is reduced by a flat amount if there is room for β. If there is no room to subtract β, the residual bandwidth is decreased multiplicatively at a rate proportional to the number of videos recently routed. This method allows connections to be more appropriately distributed when sharing a measurement snapshot.

Note that the problem formulation (Section II) does not explicitly take into account link capacity. However, capacities are considered implicitly as "*soft*" constraints every time MOS values are estimated. It is important to mention that if link capacities (and demands to $(k, l) \in S$) were explicitly considered in the problem formulation (*i.e.*, as "*hard*" constraints), the SQAPE problem would be NP-hard, as it is a generalization of the *Multi-Commodity Flow* problem [26]. Last, observe that the proposed algorithm has polynomial time complexity of $O\big(|L| + |N| \cdot log|N|\big)$ using a Fibonacci heap to extract minimum values from χ.

IV. Evaluation

This section first presents a description of the evaluation environment and its parameters in Subsection IV-A. Subsections IV-B and IV-C expand on the first research question: (i) how to accurately predict video QoE for given pairs of source and destinations in large-scale networks? Subsection IV-D addresses to the second research question: (ii) how to use the QoE indicator to dynamically select and deploy QoE-aware paths which minimize infrastructure utilization?

A. Experimental Parameters

The experimental setup[4] consists of: *(i)* a Mininet-instantiated SDN topology, based on a real LTE network deployed countrywide, with four offloading containers mirroring video contents, as shown in Fig. 3; *(ii)* a realistic video workload comprised of HTTP Adaptive Streaming (HAS) (360p, 720p and 1080p), which is applied to the topology according to each scenario's criteria over 130-minute long experiment rounds; *(iii)* microservice for each of the evaluated algorithms, responsible for selecting and deploying paths on demand; and *(iv)* an active measurement procedure to periodically measure the network state on a per-link basis.

In the performance evaluation, two distinct scenarios were considered. In Scenario 1, SQAPE is compared to a baseline solution, whose path selection is based on administrative weights, usually attributed by a traffic engineer according to a traffic matrix. This kind of static traffic engineering (TE) approach is widely employed in current networks. The weights are adjusted considering an evenly distributed load among the metropolitan layer. The Bellman-Ford algorithm is applied to determine the shortest path by link weight.

In this context, two video loads are evaluated, both approximating the maximum topology capacity. The first load conforms perfectly to the homogeneous premise on which the TE algorithm is based. The second load realistically distributes the videos, including zones with a higher concentration of demand. These different loads aim to contrast how SQAPE and TE adjust to dynamic variations of input.

[4]The full set of parameters considered in this experiment, including topology and video traces, can be downloaded from https://github.com/rtcostaf/INFOCOM2018

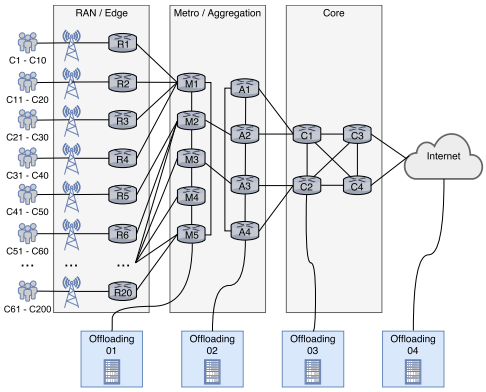


Fig. 3. Topology of the physical network considered

In Scenario 2, SQAPE is compared to four other path selection algorithms that attempt to minimize one or more network constraints regarding QoS indicators. We considered a workload composed of 744 videos, which were delimited by three components: network capacity, video bitrates and frequency of video requests (following a Poisson fit). The first contender (DKS - Delay) minimizes distance according to the Dijkstra algorithm, using delay as the distance metric [27]. This technique is commonly employed by routing mechanisms such as OSPF. The second contender (BF - BW) minimizes path's bandwidth bottleneck, using the Bellman-Ford algorithm. This strategy operates iteratively, finding the widest path with the least hops [28], [29]. The third contender is known as the Shortest Widest Path (SWP), which optimally solves the problem of maximizing TCP throughput and minimizing delay in the selected path [20]. Finally, the fourth contender is the Constrained Shortest Path First (CSPF), which is an adaptation of the Dijkstra algorithm that prunes unfeasible links before performing path calculation [11]. This algorithm is evaluated differently as it requires an assumption the other proposals do not share: it relies on the possibility of denying video delivery. Thus, it is only graphically presented in Fig. 5(c), as other representations depend on the deployment of all videos.

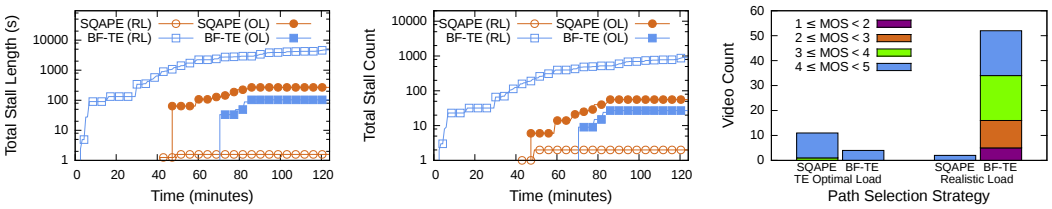
B. Video Playout

Video stall represents the most important metric to infer the quality of experience of a video playout [7], [17]. In the evaluation procedure, the player was responsible for recording stall count and duration. A stall is detected if the interval $t_i - t_{(i-1)}$ between the conclusion of the download of a video segment and that of the last segment is higher than the remaining buffer (amount of time worth of unconsumed video content available to the user) at time $t_{(i-1)}$. As one can observe in Fig. 4(a), the static traffic engineering approach, in this case an upper-bound, performed better than SQAPE in its optimal (most-favorable, yet unrealistic) load (BF-TE OL). While TE accumulated 103 seconds of stall throughout 150 videos, SQAPE stalled for 267 seconds. However, in the realistic load (RL), TE behaved considerably worse, accumulating 4,748 seconds while SQAPE stalled for 1.6 seconds. Similar results were obtained for the stall count metric (see Fig. 4(b)). Thus, SQAPE achieved competitive results considering the load specifically constructed for TE, while SQAPE outperformed TE when subjected to the realistic load.

The potential of SQAPE (w.r.t. adaptability) is more evident in Fig. 5(a) and Fig. 5(b), which present the total stall duration and stall count for Scenario 2. In such configuration, SQAPE led to 238 seconds of stall, distributed among 72 stall events. The contender solution with the closest performance (SWP) registered 204 stalls, amounting to 998 seconds of duration. The Dijkstra solution performed the worst, presenting a total of 3,815 seconds during 609 stalls. A few reasons can be given for this result. Firstly, the difference in intrinsic delays between the links causes little initial variation of selected paths (at this stage, the composition delay is mainly influenced by the propagation delay and the queue delay is minimal), which can lead to an accumulation of several videos in a few links since the early stages of the experiment. This difference can be observed in Fig. 5(a), where DKS - Delay begins to stall faster than others at around minute 25. Secondly, even though delay is a good predictor for queue occupation, the DKS - Delay algorithm does not consider available bandwidth, which is the predominant indicator for video QoE. The value of considering delay, albeit secondly, is manifested in the difference of outcome between BF-BW and SWP. Although both take the available bandwidth as a priority, SWP takes delay into account, while BF-BW minimizes hop count. This difference can be expressed in BF-BW taking paths with fewer hops but more degraded traffic queues.

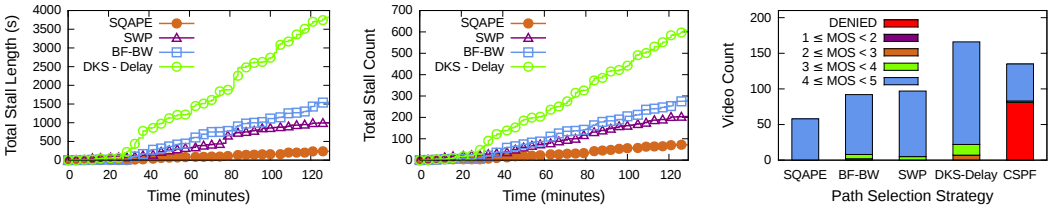
Considering CSPF separately, even though it routed 10.46% fewer videos than the other solutions, SQAPE still managed to accumulate less stall length. It should be noted that denied videos were not accounted into stall and only 7.49% of videos were responsible for all stall events in SQAPE. In light of this, it can be said that SQAPE made better use of the network resources in comparison with CSPF.

The shortest-widest path algorithm, despite performing well, did not achieve as good a result as SQAPE. SWP usually selects decent paths as available bandwidth and delay are the most influential indicators of QoE, respectively. However, the shortest-widest path is not necessarily the best one. Degradation in indicators other than available bandwidth (such as delay and loss rate) may lower QoE in the widest path. Also, an increase in available bandwidth to levels much higher than the video bitrate does not necessarily lead to an increase in QoE. Further, such an approach does not consider link utilization when determining the path. For example, SWP may prefer to select paths going through the core, where links are wider, rather than making use of local offloading containers. This tendency can overburden the network core while leaving peripheral links underused. In comparison, SQAPE considers how key QoS indicators influence the final QoE, being aware of paths which may have less available bandwidth but better overall QoE estimation. Finally, since MOS does not increase with available bandwidth indefinitely, SQAPE does not have a preference for paths wider than the expected video bitrate,



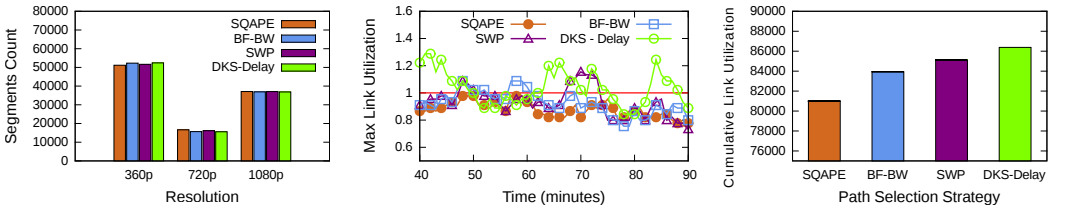
(a) Total stall length (b) Total stall count (c) Impaired videos by MOS range

Fig. 4. Scenario 1 - Video playout performance and MOS evaluation using distinct workloads (Bellman-Ford optimal vs realistic)



(a) Total stall length (b) Total stall count (c) Impaired videos by MOS range

Fig. 5. Scenario 2 - Video playout performance and MOS evaluation using realistic workload



(a) Segments count by resolution (b) Maximum link utilization histogram (c) Cumulative link utilization based on observed path length

Fig. 6. Scenario 2 - HTTP adaptive streaming performance and network efficiency using realistic workload

which allows it to make use of the network edge regularly.

C. Quality of Experience

The estimated MOS of degraded videos in Scenario 1 is shown in Fig. 4(c), considering only videos which presented some degradation ($MOS < 5$). As expected, the traffic engineered solution operated well when applied to its optimal load, resulting in only four degraded videos out of 150, all of which had MOS values considered "good" ($MOS \geq 4$). When applied to the same load, SQAPE degraded ten videos into "good" and one video into "fair" ($3 \leq MOS < 4$). On the other hand, when handling the more realistic load, the traffic engineered solution degraded more than a third of all videos, while SQAPE degraded only two. The TE solution also presented videos in the "bad" range ($1 \leq MOS < 2$), while SQAPE's worst videos were within the "good" range. This result is consistent with the stall values, confirming SQAPE's remarkable adaptability in dynamic environments.

Regarding Scenario 2, Fig. 5(c) illustrates the different amounts and intensities of degraded videos observed for each of the path selection algorithms. The number of videos denied by CSPF is also shown to compare its number of impaired video experiences. In addition to CSPF *denying* more videos than those impaired by our strategy, it still presented *impaired* videos. The DKS-Delay strategy resulted in the highest observed stall length and most impaired videos, displaying how adopting delay as the sole arbiter of quality fails to determine the best paths. Furthermore, even though BF-BW presented approximately 50% more stall length than SWP, both resulted in a similar number of impaired videos. Instead, this discrepancy was expressed in the degradation intensity experienced by the videos, evidenced by the fact that BF-BW videos reached "poor" ($2 \leq MOS < 3$), differently from SWP. All of these highly impaired videos were observed when retrieved from offloading point 1 (see Fig. 3), which is the closest content provider. This is consistent with BF-BW's preference for minimizing hop count, which represents a vulnerability depending on the network topology. SWP's auxiliary use of delay was found to be an improvement over hop count.

The performance of SQAPE achieved the lowest video degradation, registering stalls in only 58 of the 774 videos, expressing an improvement of at least 37%. Moreover, none of the stalls were intense enough to lower video MOS below

"good" ($MOS \geq 4$). In summary, despite delay being, by itself, insufficient for path selection, it is fundamental for an adequate solution. Furthermore, it is reasonable to conclude that one of the reasons SQAPE outperformed SWP is how it more appropriately incorporates delay as a metric.

Another aspect of Scenario 2 worth discussing is how the HTTP adaptive streaming influences the results. One way to evaluate such effect is to count the number of segments downloaded in each resolution, as shown in Fig. 6(a). Even though SQAPE had the edge over other algorithms (with more 1080p and 720p segments), the behavior was comparable throughout the solutions, indicating that no algorithm obtained unfair advantage in traffic demand by favoring lower resolution segments.

D. Network efficiency

According to a realistic demand, videos of up to 40 minutes of duration are requested continuously over a 90-minute window, resulting in 130 minutes of runtime. This produces three different stages. In the initial stage (for the first 40 minutes), the number of ongoing videos increases gradually, while most of the videos have not had the time to finish. The stable stage occurs between minutes 40 and 90, when the video intake is compensated by finishing videos. In the final stage (from minute 90 to 130), no new videos are requested, slowly decreasing network utilization until the last video ends.

In order to determine the efficiency of a solution with respect to network utilization, each link $(i, j) \in L$ is given an indicator $v_{ij}(t)$ that represents, at an instant t, the number of active paths the link takes part. An active path is that which has been designated and deployed for any video running at time t. Considering that the smallest resolution for Scenario 2 is 360p, there is a theoretical limit V_{ij} of how many videos can be supported by the link (i, j), defined by $V_{ij} = c_{ij}/B_{360p}$. Where c_{ij} is the link capacity and B_{360p} is the bitrate of 360p video segments, both in bits per second. When $v_{ij}(t) > V_{ij}$, considering the capacity is divided among the videos being transmitted over a link, it becomes impossible to transfer a new segment in the time it takes to be consumed. This implies the buffer is consumed faster than it grows, generating stall if and when it runs out.

Considering a network snapshot is taken at every minute, let $t \in \mathbb{N}, 0 \leq t \leq 130$. In this snapshot, value $v_{ij}(t)$ is observed for each link $(i, j) \in L$. Occupation rate $R_{ij}(t)$ is given by $R_{ij}(t) = v_{ij}(t)/V_{ij}$. The maximum link utilization at time t, designated as $ML(t)$, is defined as the maximum value of $R_{ij}(t)$ observed in all links, given by $ML(t) = max(R_{ij}(t)), \forall (i, j) \in L$.

In Fig 6(b), the evolution of $ML(t)$ is shown in the stable stage, for four of the strategies considered in Scenario 2. CSPF cannot be considered in the utilization plot because it deployed a considerably lower number of videos, not exploring the intricacies of accommodating video streams under adversity. The horizontal line indicates a theoretical limit where $v_{ij}(t) = V_{ij}$ for at least one of the links at the moment t. SQAPE was the only strategy for which the maximum link utilization remained below this threshold throughout the experiment, demonstrating how this solution makes better use of network resources.

Using the accumulated v_{ij}, one can estimate path length. By consecutively differentiating between routes deployed by each solution, it is possible to derive an indicator U of utilized links within a snapshot, given by $U = \sum_{t=0}^{130} \sum_{\forall (i,j) \in L} v_{ij}(t)$.

The value of U for each algorithm gives insight into how long are the selected paths during the experiment. Keep in mind that the longer the paths, the fewer resources are available (which could be used to transmit other contents otherwise). As shown in Fig 6(c), SQAPE had the best results in relation to the other algorithms. In a joint analysis of the plot in Fig 6(b), it can be stated that SQAPE did not require longer paths to avoid network bottlenecks. Also, solutions that made use of delay as a criterion (SWP and DKS - Delay) frequently opted for longer paths. This phenomenon can be explained by congested links presenting higher delay and paths taking a detour to avoid them, resulting in more link usage. Such paths may offer less degradation. However, higher link utilization tends to deplete the network resources more quickly, as can be seen in Fig. 6(b), where both SWP e DKS - Delay are also the ones with the worst bottlenecks $ML(t)$.

V. Related Work

The related work is organized around four main groups of investigations. The first group is characterized by techniques that perform path selection considering, among other elements, QoE indicators measured at end-user devices. QoE indicators for video streaming such as startup time, amount/duration of stalls and buffer events, when measured at end-user devices, present high accuracy, since they are obtained as close to the user as possible [9], [10], [11]. However, these techniques lead to lower system scalability, reduced flexibility and the requirement of end-user collaboration through the installation of measurement software on their devices [12]. Additionally, constraints such as privacy policies, reduced battery life and the end-users themselves, who are usually unwilling to install additional software on their device, make this group of solutions less attractive for mobile operators.

Considering the issues involved in employing end-user data, a second group of the related work makes use of server-side information, available at the content provider equipment, to estimate QoE and perform path selection [6], [13], [14]. From the mobile operator's perspective, obtaining server-side data from several video streaming providers can be a challenging task, hindering the capacity of an operator to provision paths autonomously. Additionally, neither QoE information available at server-side nor at client-side allows the operator to isolate the influence of each link on the QoE score.

The third group of related work relies on network-side information in order to estimate quality of experience and select QoE-aware paths [15], [16]. The main limitation of methods that rely on observation of video streaming data traffic, inside the network, is related to the proliferation of the HTTPS protocol. Important players such as Netflix and Youtube have adopted HTTPS as the standard protocol for

video streaming delivery. Recent research work has stated that, given the widespread adoption of HTTPS for video delivery, approaches based on streaming flow analysis became unpractical [17].

Finally, we consider the fourth group of investigations which deal with QoS-aware path selection [30], [27], [20]. These techniques do not directly consider QoE, eventually leading to an excessive emphasis on a single (or couple of) QoS metric. Variations of the Dijkstra or Bellman-Ford algorithms that consider only QoS either use unnecessarily long paths (when oriented to maximize residual bandwidth) or do not appropriately consider paths with impaired quality (when oriented to shortest paths).

VI. Conclusion

Today's decreasing ARPU and increasing video traffic challenge mobile operators to efficiently manage the interplay between infrastructure investments and user QoE. To address this issue, we proposed a novel approach comprised of QoS composition and QoS to QoE translation, which was shown to be a scalable solution for estimating video QoE in mobile networks. On top of that, we also presented a polynomial-time algorithm capable of selecting QoE-aware paths while efficiently utilizing the infrastructure resources. In a realistic evaluation comprehending mobile operator topology and video demand, SQAPE outperformed state-of-the-art solutions by impairing at least 37% fewer videos and accumulating four times less stall.

Acknowledgement

We thank the anonymous INFOCOM reviewers for their valuable feedback. This work has been funded in part by CAPES, CNPq, FAPERGS and PROPESP - IFSul.

References

[1] ITU, "Measuring the information society," International Telecommunication Union, Tech. Rep., 2016.

[2] R. Maallawi, N. Agoulmine, B. Radier, and T. B. Meriem, "A comprehensive survey on offload techniques and management in wireless access and core networks," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1582–1604, thirdquarter 2015.

[3] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," Cisco Systems, Tech. Rep., 2017.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing — a key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] N. Zhang, N. Cheng, A. T. Gamage, K. Zhang, J. W. Mark, and X. Shen, "Cloud assisted hetnets toward 5G wireless networks," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 59–65, June 2015.

[6] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng, "Engineering egress with edge fabric," in *Proceedings of the 2017 ACM SIGCOMM Conference*. New York, NY, USA: ACM, 2017.

[7] H. Nam, K. H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[8] A. Ahmad, A. Floris, and L. Atzori, "Qoe-centric service delivery: A collaborative approach among OTTs and ISPs," *Computer Networks*, vol. 110, pp. 168 – 179, 2016.

[9] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *Proceedings of the ACM SIGCOMM 2013*, 2013, pp. 339–350.

[10] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla, "SDN based QoE optimization for HTTP-based adaptive video streaming," in *Proceedings of the IEEE International Symposium on Multimedia (ISM)*, Dec 2015, pp. 120–123.

[11] H. Nam, K. H. Kim, J. Y. Kim, and H. Schulzrinne, "Towards QoE-aware video streaming using SDN," in *Proceedings of the IEEE Global Communications Conference (Globecom)*, Dec 2014, pp. 1317–1322.

[12] E. Liotou, H. Elshaer, R. Schatz, R. Irmer, M. Dohler, N. Passas, and L. Merakos, "Shaping QoE in the 5G ecosystem," in *Proceedings of the Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, May 2015, pp. 1–6.

[13] T. Uzakgider, C. Cetinkaya, and M. Sayit, "Learning-based approach for layered adaptive video streaming over SDN," *Computer Networks*, vol. 92, no. P2, pp. 357–368, Dec. 2015.

[14] A. Gangwal, M. Gupta, M. S. Gaur, V. Laxmi, and M. Conti, "Elba: Efficient layer based routing algorithm in SDN," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Aug 2016, pp. 1–7.

[15] M. Eckert, T. M. Knoll, and F. Schlegel, "Advanced MOS calculation for network based QoE estimation of TCP streamed video services," in *Proceedings of the 7th International Conference onSignal Processing and Communication Systems (ICSPCS)*, Dec 2013, pp. 1–9.

[16] A. Farshad, P. Georgopoulos, M. Broadbent, M. Mu, and N. Race, "Leveraging SDN to provide an in-network QoE measurement framework," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM Workshop)*, April 2015, pp. 239–244.

[17] P. Casas, B. Gardlo, R. Schatz, and M. Mellia, "An educated guess on QoE in operational networks through large-scale measurements," in *Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*, ser. Internet-QoE '16. New York, NY, USA: ACM, 2016, pp. 1–6.

[18] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 401–418, Firstquarter 2016.

[19] R. I. T. da Costa Filho, W. Lautenschlager, N. Kagami, V. Roesler, and L. P. Gaspary, "Network fortune cookie: Using network measurements to predict video streaming performance and QoE," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[20] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sep 1996.

[21] O. Dobrijevic, M. Santl, and M. Matijasevic, "Ant colony optimization for QoE-centric flow routing in software-defined networks," in *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 274–278.

[22] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing SDN upgrades in ISP networks," in *Proceedings of IEEE INFOCOM*, 2017.

[23] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Adaptive and reliable multipath provisioning for media transfer in SDN-based overlay networks," *Computer Communications*, vol. 106, pp. 107 – 116, 2017.

[24] T. De Pessemier, K. De Moor, W. Joseph, L. De Marez, and L. Martens, "Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching," *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 47–61, 2013.

[25] W. H. Hsu and C. H. Lo, "QoS/QoE mapping and adjustment model in the cloud-based multimedia infrastructure," *IEEE Systems Journal*, vol. 8, no. 1, pp. 247–255, March 2014.

[26] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 228 – 243, 1995.

[27] F. Kuipers, P. V. Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 50–55, Dec 2002.

[28] R. Guérin and A. Orda, "Computing shortest paths for any number of hops," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 613–620, Oct. 2002.

[29] S. Tomovic, I. Radusinovic, and N. Prasad, "Performance comparison of QoS routing algorithms applicable to large-scale SDN networks," in *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)*, Sept 2015, pp. 1–6.

[30] B. Zhang, J. Hao, and H. T. Mouftah, "Bidirectional multi-constrained routing algorithms," *IEEE Transactions on Computers*, vol. 63, no. 9, pp. 2174–2186, Sept 2014.

APPENDIX D — PAPER PUBLISHED AT ACM MMSYS 2018

- **Title**: *Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks*
- **Conference**: ACM Multimedia Systems Conference (MMSYS 2018)
- **Type**: Main track (full-paper)
- **Qualis**: A2
- **Date**: Jun 12-15 2018
- **Held at**: Amsterdam, Netherlands
- **Notes**: ACM Reproducibility Badge Award

Abstract. The demand of Virtual Reality (VR) video streaming to mobile devices is booming, as VR becomes accessible to the general public. However, the variability of conditions of mobile networks affects the perception of this type of high-bandwidth-demanding services in unexpected ways. In this situation, there is a need for novel performance assessment models fit to the new VR applications. In this paper, we present PERCEIVE, a two-stage method for predicting the perceived quality of adaptive VR videos when streamed through mobile networks. By means of machine learning techniques, our approach is able to first predict adaptive VR video playout performance, using network Quality of Service (QoS) indicators as predictors. In a second stage, it employs the predicted VR video playout performance metrics to model and estimate end-user perceived quality. The evaluation of PERCEIVE has been performed considering a real-world environment, in which VR videos are streamed while subjected to LTE/4G network conditions. The accuracy of PERCEIVE has been assessed by means of the residual error between predicted and measured values. Our approach predicts the different performance metrics of the VR playout with an average prediction error lower than 3.7% and estimates the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. Moreover, it allows us to pinpoint the QoS conditions that affect adaptive VR streaming services the most.

Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks

Roberto Irajá Tavares da Costa Filho
Federal University of RS - UFRGS
Porto Alegre, Brazil
roberto.costa@inf.ufrgs.br

Marcelo Caggiani Luizelli
Federal University of Pampa
Alegrete, Brazil
marceloluizelli@unipampa.edu.br

Maria Torres Vega
Ghent University - imec
Ghent, Belgium
maria.torresvega@ugent.be

Jeroen van der Hooft
Ghent University - imec
Ghent, Belgium
jeroen.vanderhooft@ugent.be

Stefano Petrangeli
Ghent University - imec
Ghent, Belgium
stefano.petrangeli@ugent.be

Tim Wauters
Ghent University - imec
Ghent, Belgium
tim.wauters@ugent.be

Filip De Turck
Ghent University - imec
Ghent, Belgium
filip.deturck@ugent.be

Luciano Paschoal Gaspary
Federal University of RS - UFRGS
Porto Alegre, Brazil
paschoal@inf.ufrgs.br

ABSTRACT

The demand of Virtual Reality (VR) video streaming to mobile devices is booming, as VR becomes accessible to the general public. However, the variability of conditions of mobile networks affects the perception of this type of high-bandwidth-demanding services in unexpected ways. In this situation, there is a need for novel performance assessment models fit to the new VR applications. In this paper, we present PERCEIVE, a two-stage method for predicting the perceived quality of adaptive VR videos when streamed through mobile networks. By means of machine learning techniques, our approach is able to first predict adaptive VR video playout performance, using network Quality of Service (QoS) indicators as predictors. In a second stage, it employs the predicted VR video playout performance metrics to model and estimate end-user perceived quality. The evaluation of PERCEIVE has been performed considering a real-world environment, in which VR videos are streamed while subjected to LTE/4G network condition. The accuracy of PERCEIVE has been assessed by means of the residual error between predicted and measured values. Our approach predicts the different performance metrics of the VR playout with an average prediction error lower than 3.7% and estimates the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. Moreover, it allows us to pinpoint the QoS conditions that affect adaptive VR streaming services the most.

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3204966

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Virtual reality**; • **Networks** → *Network protocols*; *Public Internet*;

KEYWORDS

Virtual Reality, HTTP Adaptive Streaming, Quality of Service, Quality of Experience, Mobile Networks

ACM Reference Format:
Roberto Irajá Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck, and Luciano Paschoal Gaspary. 2018. Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3204949.3204966

1 INTRODUCTION

The number and variety of Internet-based video applications do not cease to increase. In fact, IP video traffic is envisioned to experience a 9-fold growth between 2016 and 2021, accounting for 78% of the total mobile traffic by 2021 [5]. According to the same source, the traffic generated by Virtual Reality (VR) is expected to increase 11-fold by 2021 [5]. One key enabler for supporting such a consistent growth is the diffusion of Head Mounted Devices (HMD). HMDs are presenting high penetration rates as they (*i*) are becoming effective and affordable (*e.g.*, Google's Cardboard[1]), (*ii*) are already provided at no cost with certain smartphones (such as Samsung Gear VR[2]) and (*iii*) are being pushed by industry (*e.g.*, Facebook recently announced a permanent price drop in Oculus Go headset with the goal of reaching 1 billion VR users [3]).

In order to provide an immersive user experience, VR videos demand significantly higher bandwidths when compared to traditional video applications. These ultra-high bandwidths are not

[1]Google Cardboard https://goo.gl/DSquZf
[2]Gear VR https://goo.gl/7JdQm7

always available in wireless networks and are not easy to process by lightweight mobile devices. In fact, currently, the streaming of VR videos through mobile networks is far from optimal. A VR video contains a full 360° panoramic view, regardless of the fact that only a fraction of it, namely the viewport[3], is visible at any given instant. In an attempt to optimize bandwidth usage, a recent research path has proposed viewport-aware schemes for VR video streaming, based on the HAS (HTTP Adaptive Streaming) paradigm [10, 27]. HAS approach is focused on encoding the source content at multiple quality representations (bitrates), while each quality representation is time-segmented into small parts (*i.e.*, segments). To further optimize bandwidth usage, recent research investigations have proposed HAS variants in which quality representations are not only segmented in time but also spatially split into smaller pieces (*i.e.*, tiles) [6].

Bringing the 2D adaptive streaming techniques to the VR arena requires the VR videos to be encoded at different quality levels, temporally divided in segments and spatially tiled [19]. Then, during the streaming session, only the tiles within the viewport are streamed in high quality, while others are maintained at the lowest levels or not streamed at all [24]. To be effective, these techniques rely on viewport prediction algorithms, since the player needs to fill in a playout buffer with tiles that are expected to compose the viewport in the near future [23].

Although the use of viewport-aware techniques leads to the reduction of bandwidth consumption, the effects of network performance on VR video streaming still plays an important role on the user's perception of the services. Since the full panoramic view of a VR video usually demands a much higher bitrate, when compared to regular videos [7], even a fraction of it (viewport) may require high bitrates. Along these lines, recent investigations have emphasized the importance of the network effects on the perceived quality (Quality of Experience, QoE) of adaptive video streaming applications [8, 10, 13, 14, 16]. However, state-of-the-art approaches fall short in predicting the perceived quality for VR videos as they do not consider the spatial segmentation.

QoE has shown to be a critical factor for video applications [1, 18]. As such, both network operators and VR content providers are required to answer an important question: *considering the wide range of performance levels of IP networks, to which extent are the currently observable network conditions able to provide users of VR applications with adequate QoE?* Answering this question is remarkably complex due to two constrains. First, the influence of the network on VR video performance is unknown; and second, the state-of-the-art on video QoE estimation modeling does not consider the VR context.

In this paper, we present PERCEIVE (PERformanCe EstImation for VR vidEos), a method that aims to provide answer to both aspects. PERCEIVE is a two-stage adaptive VR performance assessment model that employs machine learning algorithms to first predict VR video playout performance, using network QoS indicators as predictors. Then, it uses the video playout performance metrics to model and estimate the end-user perceived quality. Evaluated in real-world 4G/LTE network conditions, PERCEIVE not only accurately predicts the VR videos performance over networks, but also allows us to pinpoint the QoS conditions that affect VR streaming services the most.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce the approach used for the tile-based adaptive VR video streaming. In Section 4, we describe PERCEIVE, the two-step performance prediction scheme that we propose. In Section 5, we report the evaluation carried out to prove concept and technical feasibility of the proposed approach. It includes details on the evaluation methodology, the generation of the dataset, analysis of the training set and results. Finally, our conclusions and key findings are presented in Section 6.

2 RELATED WORK

In this section, we provide a thorough description of the state-of-the-art. In Section 2.1, we present a brief introduction to adaptive streaming applied to the VR context and review the state-of-the-art approaches. In Section 2.2, we highlight the most significant QoE estimation models in literature.

2.1 Adaptive Tile-based and Viewport-aware Video Streaming

An aspect to consider in adaptive tile-based VR streaming services is viewport prediction, which allows to considerably optimize bandwidth usage. Since a full VR video can easily reach 8K video resolution [7], most video players rely on heuristic algorithms to predict near-future user's head movements. By considering next position prediction, the video player is able to request only tiles that are likely to be inside the viewport, which leads to reduced bandwidth utilization. To provide this prediction, heuristic algorithms consider variables such as the angular velocity of the user's head, movement patterns for previous viewers, video content (*e.g.*, in a football match users will most likely follow the ball's movements), among other factors [2]. By performing such prediction, the video player can reduce in up to 72% in bandwidth utilization [12].

In practice, the viewport prediction algorithm is responsible for keeping a small video playout buffer (*e.g.*, 2 seconds) with the tiles that are more likely to belong to the viewport in the near-future. To illustrate how the viewport prediction interacts with the playout buffer, consider the example of a user watching a tile-based VR video using a head-mounted display. Consider a given temporal segment S_k and a respective viewport V_k, as depicted in Figure 1 (a). At this moment, the video player is requesting high-resolution chunks only for tiles inside the viewport V_k. Then, based on the near-future viewport prediction for the next segment (S_{k+1}), the video player starts requesting high-resolution tiles for the viewport V_{k+1} (delimited by the right dashed square in Figure 1 (b)). As predicted, the viewer slightly moves to the right (see Figure 1 (c)). At this point, driven by the viewport predictor, the VR player starts requesting high-resolution chunks for the predicted viewport on the segment S_{k+2} (Figure 1 (d)), and so forth.

Several recent investigations [11, 23, 24] have focused on a common main objective: devising bandwidth-efficient adaptive VR video players while keeping QoE at acceptable levels. Taking viewport prediction information as input, most approaches rely on per-tile rate adaptation algorithms to reduce the amount of information to

[3] Also referred to as Field of View (FoV)

be downloaded by keeping only the viewport's tiles in high resolution. Qian *et al.* [24] present a viewport prediction scheme that considers users' head movements (traces) and relies on Weighted Linear Regression to predict users' head position for the next second. The same study indicates that the estimation accuracy can drop from 90% to approximately 60% when increasing the prediction window to 2 seconds. Fan *et al.* [11] consider HMD sensor information and content-related features (*i.e.*, image saliency maps and motion maps) to train two neural networks for prediction of viewport position. In turn, Hosseini *et al.* [23] propose an efficient 3D geometry mesh to represent tiled VR video in the 3D space, which is capable of reducing bandwidth consumption in up to 30% when compared to non-tiled videos. The work also relies on MPEG-DASH Spatial Representation Descriptor (SRD) to describe the spatial relationship of tiles in the 360-degree space, prioritizing tiles inside the viewport to be downloaded in high quality. Petrangeli *et al.* [23] propose an approach with the ability to reduce bandwidth consumption in up to 35% by relying on both an HTTP/2-based push mechanism and a viewport prediction scheme based on viewport speed.

As discussed, viewport prediction is a sensitive task, which might affect the user's perception in unexpected ways. Errors on the prediction of the viewport (*i.e.* the FoV that the user will look at in the next segment) may lead to partial or full degradation of the perception, even if the network conditions are enough to guarantee the user's QoE. This means that, during the streaming, two different processes (namely the viewport prediction and the effects of the network on the adaptive streaming performance metrics) will have a major influence on the user's QoE. In this work, we are interested on predicting the effects of networks on VR adaptive streaming in an isolated manner, without the influence of errors derived from wrong viewport prediction. Thus, for the analysis presented herein, we have assumed perfect prediction, *i.e.*, the adaptive streaming player knows exactly where the user is looking at every point in time.

2.2 Adaptive Video Streaming QoE Estimation

One of the biggest challenges of adaptive video streaming (2D, 3D or VR) applications is the accurate and real-time estimation of quality perceived by the users. And, based on it, the provision of a feedback loop to dynamically influence the quality adaptation. In state-of-the-art adaptive streaming approaches, the modeling of QoE has to rely on objective information obtained at the client, the server or the network-side.

Mao *et al.* [16] present a model to estimate QoE considering both network and application performance indicators measured at client's video player (*e.g.*, average bitrate, video stall events and bitrate changes) as inputs. Similarly, Jiang *et al.* [13] propose a Content Delivery Network (CDN) node selection approach that employs a Critical Feature Analytics (CFA) design to provide accurate QoE estimation. In this work, the authors also rely on information provided by client video players as input. Conversely, Xianshu *et al.* [14] propose a network path selection scheme that considers the bitrate measured at the server-side to produce simplified QoE estimation for adaptive video streaming applications.

Dimopoulos *et al.* [10] introduce a methodology for estimating QoE based on the analysis of encrypted HTTPS video streaming traffic observed in the network. Da Costa Filho *et al.* [8] propose an approach for QoE estimation based on network QoS indicators obtained through active measurements. Both investigations [8, 10] demonstrate it is possible to estimate video streaming QoE based on network-side information. Although these approaches may not be as accurate as the ones based on client- or server-side measurements, they have shown to result in a satisfactory level of accuracy with a crucial advantage: in addition to end-to-end QoE estimation, they allow for fast identification and isolation of network segments responsible for QoE impairments. In spite of the recent fundamental contributions for the video streaming evolution, the work on QoE modeling for adaptive video streaming has basically focused on 2D videos. Unlike 2D video content, VR presents significantly more complex elements to consider (*e.g.*, spatially segmentation, viewport prediction, and per-tile rate adaptation). Thus, the current QoE models are not suitable to estimate QoE for VR videos.

3 ADAPTIVE STREAMING OF VR VIDEOS USING TILES AND QUALITY ZONES

This section introduces the adaptive VR streaming approach for which PERCEIVE is envisioned. In order to reduce the bandwidth required for the streaming, it adopts a tiling structure, in which the videos are not only divided in temporal segments but are also spatially split in sections (tiles) [23]. In addition, tiles are grouped in quality zones prior to the streaming. Each of the zones is assigned a quality level according to the network conditions measured during the previous segment. In the next two subsections, both the structure and the adaptive streaming technique adopted in this work are presented.

3.1 Adaptive VR streaming structure: Spatial Tiles and Quality Zones

A VR video V can be represented by a set of k spatially divided zones $Z = \{Z_1, ..., Z_k\}$ such that $\bigcap_{\forall k} Z_k = \emptyset$. The same video V is temporally split into a discrete number of m segments $S = \{S_1, ..., S_m\}$ such that $\bigcup_{\forall m} S_m = V$. Each zone Z_k is composed of a set of tiles $t \in Z_k$. A tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$, and may assume different bitrates (qualities) $R(C_{t_m})$ over time. Finally, we refer to a segment as the set of all chunks for a given time frame such as $S_m = \bigcup_{\forall t} C_{t_m}$. In tile-based approaches, the encoding process defines how the video will be spatially divided (*i.e.*, tiling scheme), which bitrates will be available in the HAS context (*i.e.*, quality representations), and the segment length (*i.e.*, number of seconds).

An example of this type of structure is shown in Figure 2. There are three quality zones $Z = \{Z_1, Z_2, Z_3\}$, each one composed by a set of adjacent tiles. Z_1 is a set of tiles adjacent to the viewport center ($t_{28}, t_{29}, t_{36}, t_{37}$), Z_2 is the border of the viewport ($t_{43}, t_{44}, t_{45}, t_{46}, t_{38}, t_{30}, t_{22}, t_{21}, t_{20}, t_{19}, t_{27}, t_{35}$) and Z_3 is composed by all tiles outside the viewport.

3.2 Adaptive streaming heuristic

Algorithm 1 describes the adaptive streaming heuristic procedure adopted for this paper (adapted from [23]). The bitrate in a specific zone Z_k is named as $R(C_t)|^{Z_k}$. The algorithm receives as input *(i)*

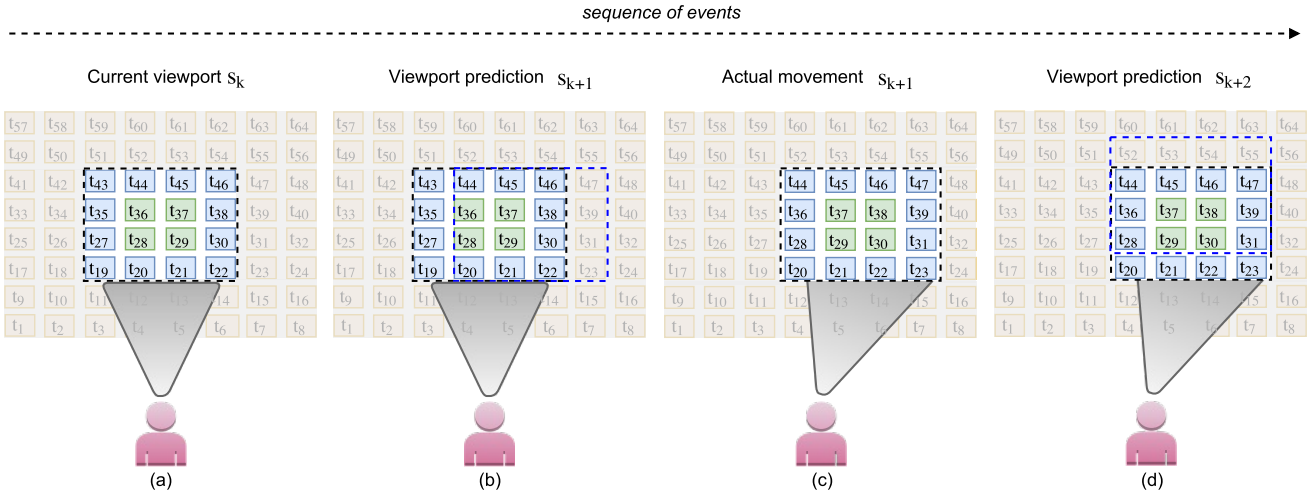


Figure 1: Example of the working principles of the viewport prediction.

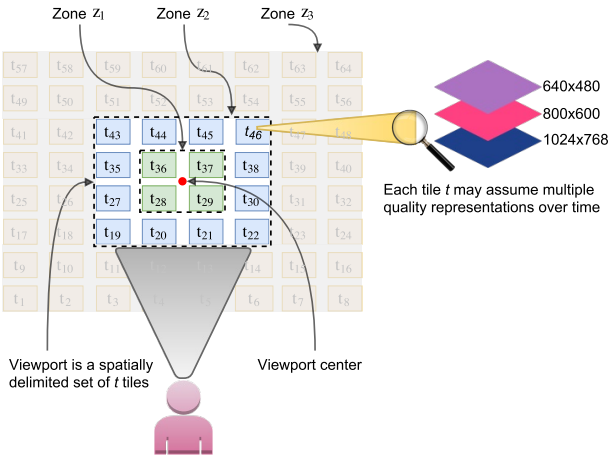


Figure 2: Example of an adaptive tile-based VR video structure split in 3 quality zones.

a reference to a VR video V, *(ii)* a set S describing the video segmentation and *(iii)* the available zones in video V. The heuristic described in Algorithm 1 works as follows. Once knowing the available bandwidth in the network, the VR player downloads tiles with the highest possible bitrates. First, the heuristic tries to increase the bitrates on the zones inside the viewport. Then, it repeats the procedure to stream tiles from the outer zones (observed in line 3). Observe that the heuristic does not increase the bitrate $R(C_t)|^{Z_{k+1}}$ on a subsequent zone Z_{k+1} in case the bitrate of zone Z_k is strictly upper-bounded by $R(C_t)|^{Z_{k+1}}$. In other words, it ensures that the bitrate of zone Z_{k+1} is always lower or equal to that of zone Z_k. Furthermore, it ensures that tiles within the same zone Z_k are streamed with the same bitrate $R(C_t)$, that is, $R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$. If the available bandwidth were insufficient to download all the tiles in a zone on time (before display), the streaming would stall until the buffers were filled. Hence, the player ensures that all tiles are synchronized during the playout and that no black tiles appear. For more information on the working principles of the streaming heuristic, please refer to [23].

Algorithm 1 VR player heuristic adapted from [23].

Input: V: VR video
Input: S: discrete number of segments in VR video V
Input: $Z = \{Z_1, ..., Z_k\}$: k spatially divided zones in VR video V
1: **for** each video segment $S_i \in S$ from VR video V **do**
2: **for** each zone $Z_k \in Z$ **do**
3: *gather* tiles $t \in Z_k$ with maximum available bitrate $R(C_t)|^{Z_k}$, such that $(\forall k \geq 2) : R(C_t)|^{Z_k} \leq R(C_t)|^{Z_{k-1}}$ and $(\forall t \in Z_k) : R(C_0)|^{Z_k} = R(C_1)|^{Z_k} = \cdots = R(C_t)|^{Z_k}$

4 PERCEIVE: ADAPTIVE VR VIDEO PERFORMANCE PREDICTION

Figure 3 presents the block diagram of the proposed two-stage VR performance prediction method. The first stage is composed of four predictors, one per VR video application performance metric (*i.e.*, startup delay, quality, quality switches count and video stalls) [32]. As input, the predictors consider both the network Quality of Service (QoS) (*i.e.*, delay, packet loss and TCP throughput) and the tiling scheme. In the second stage, the user QoE is estimated by submitting the predicted application layer performance metrics to the proposed QoE model. PERCEIVE can dissect VR video playout performance by understanding two key processes, namely *(i)* the influence of the network performance on VR video player outputs; and *(ii)* how the user perceives the resulting video playout performance. However, both the VR video player and the QoE model are open questions in the sense that there is neither a reference player implementation nor a QoE model for VR videos. Considering the above, the proposed two-stage prediction allows both the playout

performance metrics predictors and the QoE model to be individually updated, without the need to rebuild the entire scheme. The following two subsections provide details and insights on each of the stages of which PERCEIVE is composed.

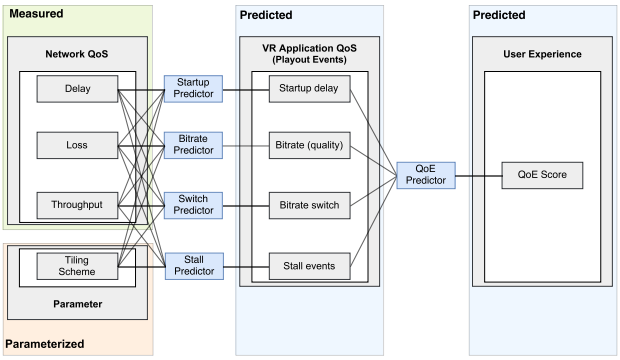


Figure 3: PERCEIVE two-stage quality prediction

4.1 Adaptive VR Video Playout Prediction

In the first stage of the method, the four most important playout performance metrics associated with the adaptive VR streaming, namely startup delay, quality level (bitrate), quality switches and stall time [32], are predicted based on network QoS inputs and the VR video structure. Each of them is independently predicted using regression trees as predictors (taking advantage of and adapting the 2D procedure proposed in previous research work [8]). Regression trees are employed due to three main reasons. First, they have shown to be an accurate machine learning method in related investigations [8, 26]. Second, they permit understanding complex and non-linear relationships between predictors and response variables in an intuitive and visual manner. This is a very important feature that allows to pinpoint the most influential inputs, which will be put to the test in the analysis of Section 5.3. Finally, once the prediction structures are generated, they can predict the response variable in linear time complexity, and can be easily integrated in third-party applications, which are fundamental aspects for network operators and content providers.

The selection of the input QoS parameters has been made based on state-of-the-art research studies on the QoS conditions that affect video streaming services the most [8, 20, 28]. These studies also concluded that TCP throughput is one of the most influential QoS metric when prediction QoE. Also, both network losses and delays have been demonstrated to be responsible for further degradation, depending on the type of streaming application used. In addition to these three network performance metrics, a fourth parameter, namely the tiling structure of the VR streaming, was included. The structure defines the number of tiles that need to be streamed to the client, thus it will heavily influence the VR playout performance. Once the four VR playout performance indicators are predicted, they serve as input to the second phase, the QoE model as it is presented in the next Section.

4.2 Adaptive VR QoE Estimation Model

The purpose of this second stage is to estimate the quality perceived by the end users (their QoE), considering the VR video playout performance metrics obtained from the previous stage of PERCEIVE (*i.e.,* startup delay, quality level (bitrate), quality switches and stall time).

The model proposed herein considers the state-of-the-art on QoE modeling for adaptive streaming applications in general and HAS in particular [16, 22, 32]. To the best of our knowledge, this is the first model to consider the concept of zones and tiles in a QoE estimation model for VR videos. These characteristics are crucial, given the fact that they allow coping with VR video attributes while providing flexibility to handle different video encoding strategies (*e.g.*, tiling scheme, viewport geometry and available quality representations).

Given the concepts of quality zones and tiles of the approach used for Adaptive VR streaming (Section 3), the QoE function is defined per zone as a function of the VR playout characteristics predicted by the previous stage within that quality zone (Section 4.1). This strategy is aligned with the notion that the influence of VR playout characteristics on user perception is different depending on the zone where they are observed (*e.g.*, quality switches for tiles outside the viewport are less important than quality switches inside the viewport). Thus, the per-zone quality function ($\phi(Z_k)$) is defined as the weighted sum of the four playout characteristics (Equation 1).

$$\phi(Z_k) = \overbrace{\sum_{\forall t \in Z_k} \sum_{\forall c \in C(t)} q(R(C_{t_m}))}^{\text{Quality}} - \overbrace{\mu \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+}^{\text{Stalls}} - \underbrace{\lambda \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left| q(R(C_{t_m+1})) - q(R(C_{t_m})) \right|}_{\text{Quality switches}} - \underbrace{\omega \cdot T_s}_{\text{Startup}} \quad (1)$$

In Equation 1, $R(C_{t_m})$ R represents the bitrate (*i.e.,* quality) of a given chunk. Recall that a tile t is time-divided into m chunks $C = \{C_{t_1}, ..., C_{t_m}\}$, (adapted from [16, 32]). Function q is a mapping function that translates the bitrate of chunk C_{t_m} belonging to tile t to the quality perceived by the user (*i.e.*, in terms of bitrate sensitivity). The second term of the Equation is used to track stall time. Stalls can be characterized either by tile (*i.e.*, it is possible to have stall in some tiles and video playout in other, for the same segment) or by segment (*i.e.*, the video will stall until all the tiles for a given segment have enough buffer). To keep the model as general as possible, we consider for each chunk c, that a stall event occurs when the download time $\frac{d_c(R_c)}{C_c}$ is higher than the playout buffer length (B_c) when the chunk download started. Hence, the total stall time is given by $\sum_{c=1}^{C} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+$. In addition, $|q(R_{ct+1}) - q(R_{ct})|$ considers the quality switches between consecutive chunks and T_s tracks the startup delay. Finally, constants μ, λ, ω are the non-negative weights used to tune the model for different user importance regarding QoE events. For example, a higher value of μ, with respect to the other weights, means that the user is more

susceptive to video stalls. Consequently, these events should affect the QoE indicator more severely.

Each of the zones within the VR video influences the perception of the user in a different manner. For example, tiles within the first or second zones (*i.e.*, the closest to the FoV of the user) will greatly steer the quality perceived by the user, while bad qualities on tiles of the edge zones will potentially go unnoticed. For this reason, the overall video QoE ($\phi(V)$) is modelled as a weighted linear sum of the QoE measurement per zone (Equation 2). Each weight ($\alpha_1, \alpha_2, ..., \alpha_k$) allows defining the relative importance of each zone when composing the video QoE. For example, the zones belonging to the viewport should have higher weights compared to the other zones.

The values for each α_n parameter should be derived from subjective tests. For example, considering a two-zone QoE scheme, values for α_1 (viewport) should be close to one, and values for α_2 (outside viewport) should be close to zero. When the QoE model is configured with more than two zones, it is necessary to determine α_n (testing values within a certain range) for each zone. In this case, subjective tests should systematically include incremental quality degradation, specifically in the intermediate zones, in order to measure the user's sensitiveness regarding quality issues in each zone.

$$\phi(V) = \alpha_1 \cdot \phi(Z_1) + \alpha_2 \cdot \phi(Z_2) + ... + \alpha_k \cdot \phi(Z_k) \quad (2)$$

5 EVALUATION

In this section, the evaluation of the PERCEIVE method is discussed. We start by presenting the procedure followed to evaluate the method in Section 5.1. Next, in Section 5.2, we introduce the generation of the dataset used for training and testing. In Section 5.3 we discuss and analyze the resulting VR playout predictors. This analysis provides insights on the dependency and predictability of each of the VR playout performance metrics given the QoS and tiling structure inputs. Finally, in Section 5.4 we present the prediction evaluation results for each of the five outputs of PERCEIVE (*i.e.*, the four VR playout performance metrics and the perceived quality).

5.1 Evaluation Methodology

In order to evaluate the performance of PERCEIVE, the procedure outlined by Figure 4 is followed. First, the datasets for training and testing have to be generated. Therefore, a VR video player is required to measure the VR video application playout performance metrics (*i.e.*, startup delay, average bitrate (quality), bitrate switches and video stalls) while subjected to real-world inputs, such as a realistic wireless networks measurements, VR tile-based videos and users' head track traces.

Next, the resultant datasets are given as input to the machine learning algorithm (responsible for learning the influence of the network QoS parameters and tiling scheme onto the VR playout characteristics). After the training phase, the resulting predictors can estimate the application layer performance only by means of the network parameters, and the considered tiling scheme. Finally, based on the VR playout performance metrics, the QoE can be

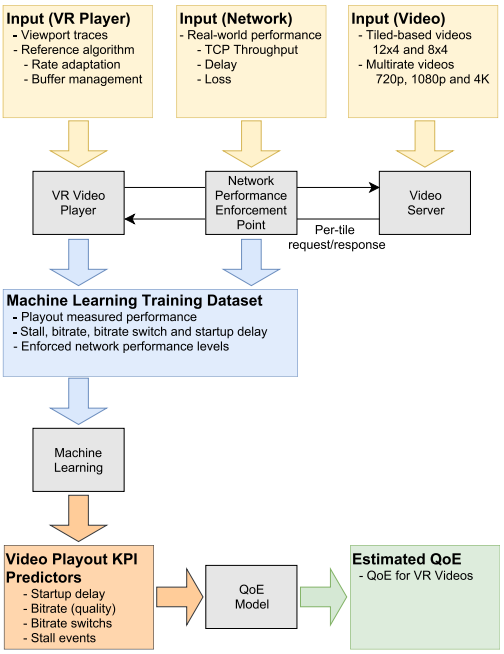


Figure 4: General evaluation methodology for PERCEIVE

estimated. The performance of PERCEIVE is assessed by means of the calculation of the normalized residual errors between predicted and measured values (r_i, Equation 3). In the equation, x is the ground truth, $\hat{x}$ is the prediction and N is the normalization factor (in this case the video duration).

$$r_i = |\hat{x}_i - x_i|/N \quad (3)$$

5.2 VR Dataset Generation

Each sample in the dataset contains the VR video tiling information, the three network QoS features and the respective video performance measured by the VR video player. To construct such dataset, the procedure presented in Section 5.1 is followed. Experiments are set, considering that a VR video player requests and processes tile-based VR videos from a web server (Apache 2 2.4.18-2). The network conditions are enforced by the Linux Traffic Control (TC) mechanism according to real-world network performance inputs. The experiments are built on top of a Linux Ubuntu 14.04 operating system, running on bare metal servers, where each server consists of a quad-core E3-1220v3 (3.1GHz) processor, with 16GB of RAM and two 10-gigabit network interfaces. Considering this infrastructure setup, we performed 1,524 video execution rounds, which resulted in more than 5,240 minutes of VR video playout.

Table 1 summarizes the input parameters values considered in the experiments. As network throughput input, the 4G/LTE measurements dataset of van der Hooft *et al.* [29] was selected. This dataset presents TCP throughput ranging from 0 Kb/s to 95 Mb/s as shown in Figure 5. For network packet loss, values between 0% and 13% were selected, in line with [8]. The network delay range was set from 1 to 130 ms. These values allowed us to assess the

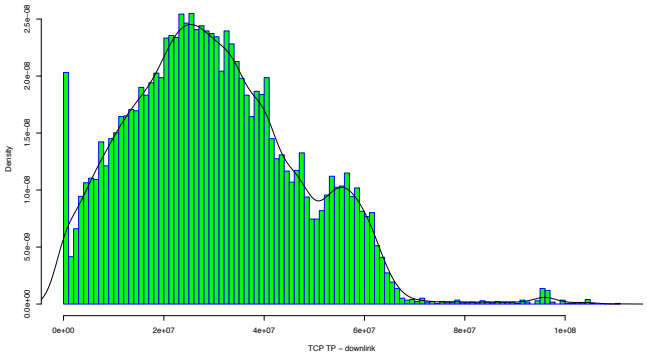


Figure 5: TCP throughput histogram of the 4G/LTE dataset of [29]

application performance from a very degraded delay performance (130 ms) down to the expected 5G delay (1 ms) [9].

Table 1: Input parameter configurations.

Metric	Short	Unit	Range
TCP throughput	TCPTP	Mb/s	0-95Mb/s ([29], Figure 5)
Packet Loss	PLR	%	0 – 13% (based on [8])
Delay	Delay	millisecond	1-130ms (based on [8, 9])
Tiling scheme	Tile	categorical	8×4 or 12×4 (based on [15, 24]

Two VR videos from Wu et al.' s dataset [31] (namely "Google Spotlight-HELP" and "Freestyle Skiing") were used for the streaming under the above described network conditions. For each video, we considered the available datasets regarding the users' head movements while watching it. As the original videos are not tile-based, they had to be re-encoded. After extracting the raw YUV files, making use of the Kvazaar encoder [30], the videos were re-encoded in a HEVC tile-based version, considering two tiling schemes: 8×4 and 12×4 [15, 24]. In addition, each tiling scheme was encoded to three quality representations, namely 720p (1.8Mb/s), 1080p (2.7Mb/s) and 4K (6Mb/s). Next, we used the MP4Box[4] application to pack the re-encoded videos into MP4 containers. Finally, we defined the segment duration of 1 second and used MP4Box to extract per-tile files and to generate the MPEG Dash Media Presentation Description (MPD) files considering multiple quality representations (Table 2). For the streaming heuristic (Section 3.2), there are three defined zones, where Zone 1 is the viewport center tile, Zone 2 groups the 8 tiles surrounding Zone 1, and all other tiles belong to Zone 3. Figure 6 shows the zone division for the 12×4 tiling scheme.

Table 2: Adaptive streaming configurations.

Videos	Qualities (bitrates)	Quality zones	Segment	Tiling
Google Spotlight Freestyle Skiing (Wu et al. [31])	720p - 1.8Mb/s 1080p - 2.7Mb/s 4K - 6Mb/s	Zone 1: 1 tile (central FoV) Zone 2: 8 tiles (adj. Zone 1) Zone 3: Rest	1 s	12×8 8×4

[4]MP4Box https://gpac.wp.imt.fr/mp4box/

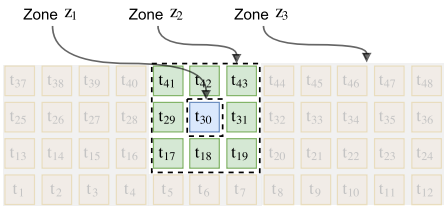


Figure 6: Viewport detail for the 12x4 tiling scheme

5.3 Resulting Predictors: VR Playout vs Network Conditions

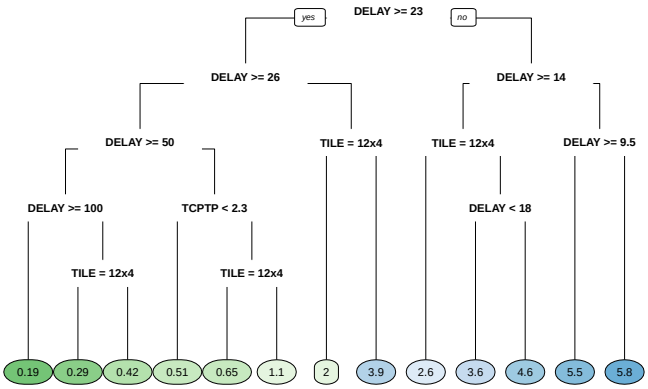
Based on the dataset, the regression trees were trained using a 10-fold cross-validation approach [15]. As each zone has independent quality behavior, both the quality and quality switches need to be learned per-zone. On the other hand, the startup and stall times are independent from the quality zone under scrutiny. Hence, they can be learned per video segment. Given the fact that there are three quality zones, eight regression trees were trained: three for Quality, three for Quality Switches, one for Stall time and one for Start time. All trees are optimally pruned [17], which means pruning until the cross-validation error is minimal and overfitting is avoided.

Before assessing the performance of the two-stage method, a thorough analysis of the regression trees was performed. This analysis aims at characterizing the relationship between the input parameters (network conditions and VR video structure) and the VR playout, allowing one to pinpoint to the most influential inputs.

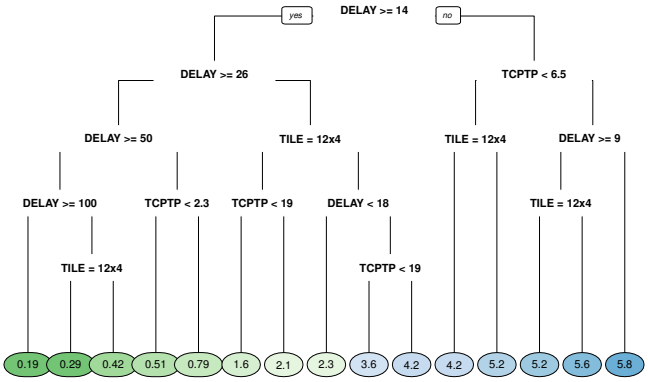
Figures 7 to 9 present the outcome predictors derived from the regression trees. All presented trees share two structural characteristics. First, although inversions may occur, usually the leftmost leaf node holds the lowest value for the predicted variable, and the value increases while moving towards the rightmost leaf node. Second, the closer to the root node, the more important the prediction feature (*i.e.*, delay, TCP throughput, loss and tile scheme).

Having a first look at the content of the trees, two observations can be made. First, network packet losses are not included in any of the trees. This means that the level of packet losses does not have influence on the VR playout performance metrics. Its effect will only be important as they affect the TCP throughput (higher network packet losses = lower TCP throughput). Furthermore, network delays turn out to be the most influential parameter on the VR playout.

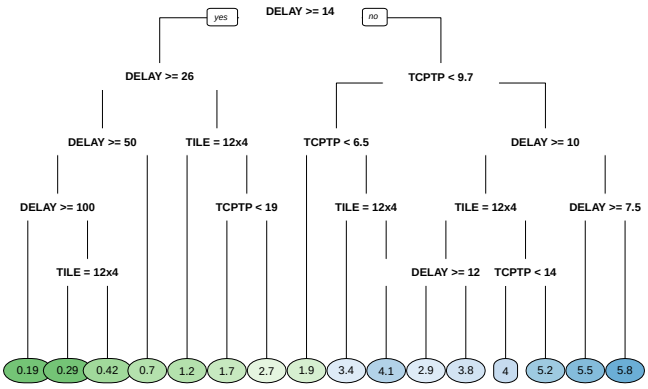
Regarding quality (by means of the average bitrate) (Figures 7(a) to 7(c)), let us consider the following aspects. The first decision taken in Zone 1, at the root node and, therefore, the most influential, is to understand if the network delay is greater than 23 ms (Figure 7(a)). The left branch ($Delay \geq 23ms$) is related to predicted quality not higher than 3.9 Mb/s, regardless of any other input value. In other words, even considering that the evaluated LTE network presents TCP throughput of up to 95 Mb/s, it is not enough to achieve the maximum bitrate (6.0 Mb/s - 4K), if the delay is higher than 23 ms. The reasoning behind this behavior is that each video segment (1 s) demands the download of 32 (8x4 tiling) or 48 (12x4) tiles. Despite the reuse of the TCP connection avoids the TCP slow-start restart [4]), the request/response overhead limits the throughput.



(a) Quality (average bitrate) - Zone 1 (Mb/s)

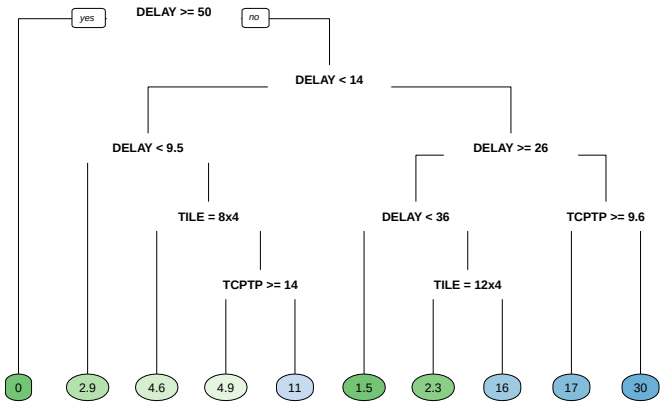


(b) Quality (average bitrate) - Zone 2 (Mb/s)

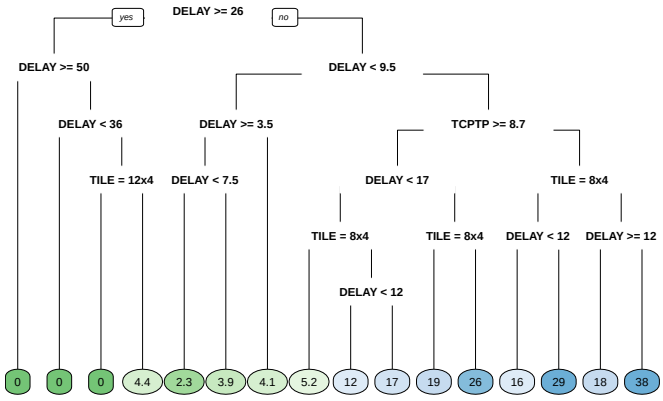


(c) Quality (average bitrate) - Zone 3 (Mb/s)

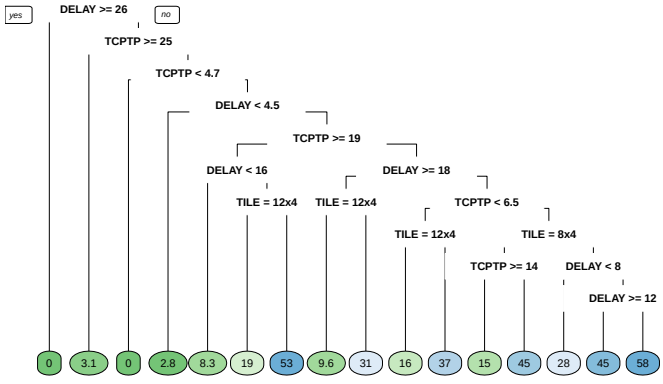
Figure 7: Regression tree representation for the predictors of the VR playout performance metric: quality. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).



(a) Quality switch - Zone 1



(b) Quality switch - Zone 2



(c) Quality switch - Zone 3

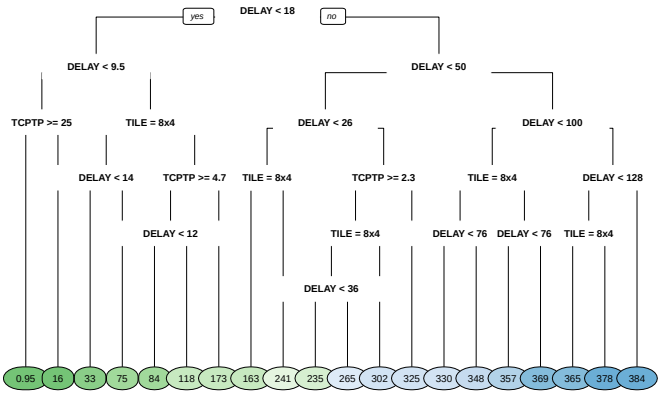
Figure 8: Regression tree representation for the predictors of the VR playout performance metrics: quality switches. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).

For Zones 2 and 3 (Figures 7(b) and 7(c)), the quality predictors follow a very similar trend. However, in order to achieve the same level of average quality, they demand higher network performance than Zone 1. The right-most leaf of Zones 2 and 3 are a clear example of this behavior. To achieve the same quality (average bitrate of 5.8 Mb/s), Zone 2 requires a delay lower than 9 ms, and Zone 3 lower than 7.5 ms. Also, the values of TCP throughput to achieve intermediate average bitrates are higher for Zones 2 and 3 when compared with Zone 1. The main reason for such behavior comes from the rate adaptation heuristic, which prioritizes high bitrates for the tiles that are closest to the viewport's center (Section 3.2). Thus, intermediate network performance may be enough to keep Zone 1 at the highest available bitrate, while high levels of network performance allow increasing the bitrate for all zones.

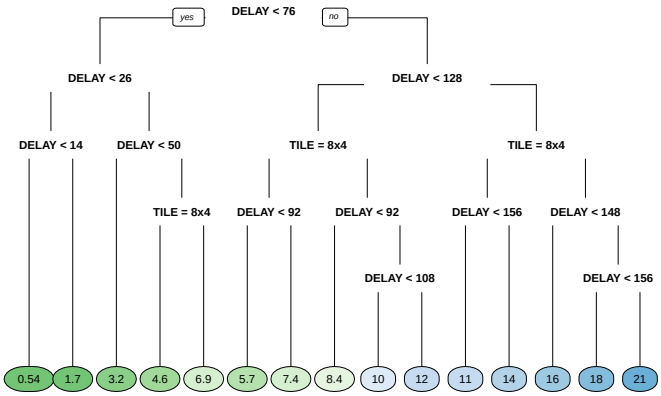
Quality switches (Figures 8(a) to 8(c)) provide valuable information in the context of HAS videos. For example, if no switches occur, the full video playout occurs in the lowest available resolution, meaning that the video player is unable to switch to higher bitrates, probably, due to insufficient network performance. In turn, when subject to excellent network performance conditions, most of HAS rate adaptation heuristics (including the one used in this paper) will stabilize at the highest available bitrate within a few switches. When considering real-world networks, if we have a look at the quality switches trained trees (Figures 8(a), 8(b) and 8(c)), it can be seen that the turning point from zero switches to maximizing the quality is a network delay of 50ms for Zone 1, and 26ms for Zones 2 and 3. However, by analyzing the rightmost leaf nodes of the decision trees for Zones 1, 2 and 3, one can observe that the maximum number of quality switches increases from Zone 1 towards Zone 3: (30, 38 and 58, respectively). This happens because, according to the considered heuristic for rate adaptation, the tiles inside Zone 3 will be the first ones to be switched to a lower resolution in case a network performance degradation is detected, followed by Zone 2 and, if the network performance degradation is severe, the Zone 1.

With respect to the cumulative stall time (Figure 9(a)), the resulting regression tree presents a wide range of predicted values (from 0.95 up to 384 seconds). One key aspect is related to the decision taken at the root node. As one can observe, if the delay is higher or equal to 18 ms, the minimum expected stall time is equal or higher than 163 seconds, independent of the tiling scheme or the available bandwidth (TCP throughput). Such high values would inflict a dramatic degradation on the perceived quality. In turn, for network delays lower than 9.5 s and TCP throughput equal or higher than 25 Mb/s, the expected stall time is minimal (0.95 seconds). It is worth mentioning that, even if the delay is lower than 9.5 s, if the TCP throughput is lower than 25 Mb/s, the expected stall time is 16 seconds. Also, in line with the aforementioned findings, the 12x4 tiling scheme leads to a significative higher amount of stall time for intermediate levels of network performance.

Finally, the regression tree for predicting startup delay is shown in Figure 9(b). In the considered VR video player, the startup delay is characterized as the elapsed time between the arrival of the request for the first tile and the completion of the buffer filling for all tiles for the first two segments. As the segment is relatively small, and considering the small file size of the tile chunks (on average 23 KB for 4K video resolution), the startup delay exclusively depends on



(a) Cumulative start time (seconds)



(b) Startup delay (seconds)

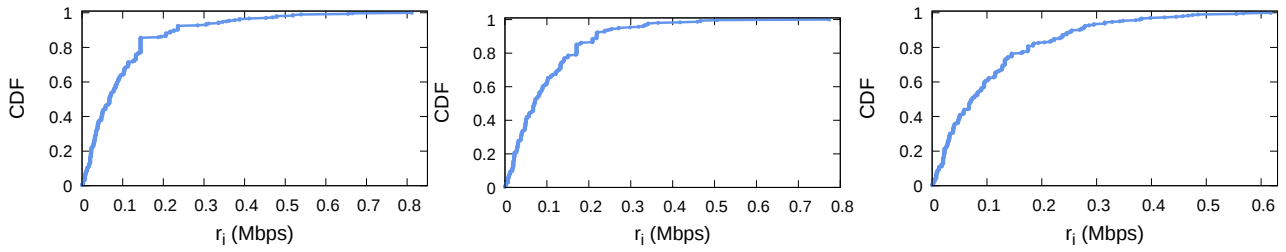
Figure 9: Regression tree representation for the predictors of the VR playout performance metrics: stall time and startup delay. Leaf node colors go from dark green (for the lowest value for the predicted variable) to dark blue (the highest predicted value).

the network delay. A delay lower than 26 ms is enough to provide an acceptable startup delay (smaller than 1.7 s). However, the best performance is achieved when the delay is lower than 14 ms (0.54 s).

5.4 PERCEIVE Results

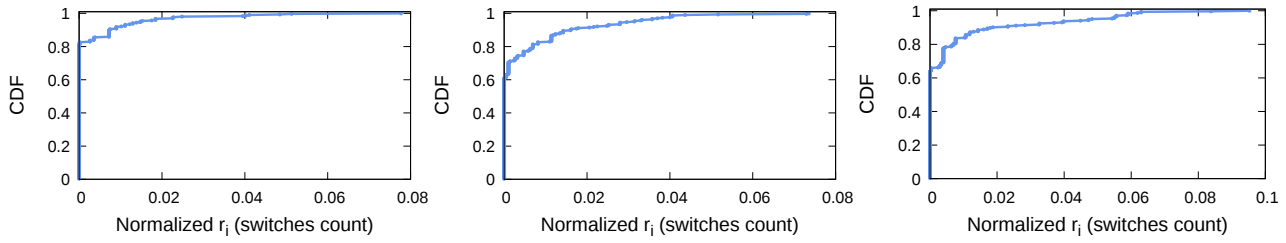
Aiming at determining the accuracy of the proposed predictors, the trained regression trees were used on unseen samples of the generated dataset, according to a 10-fold cross-validation scheme [15]. We considered as ground truth the performance measured by the reference VR video player when subjected to real-world network performance traces. In light of this, each test sample i contains the predictor variables (*i.e.*, TCP throughput, delay and tiling scheme), and the respective measured values for the performance metrics (*i.e.*, average bitrate, stall time, quality switch and startup delay).

Furthermore, based on the predicted VR playout characteristics, the QoE indexes were estimated by means of Equation 2. The parametric constants shown by the model were set to the values



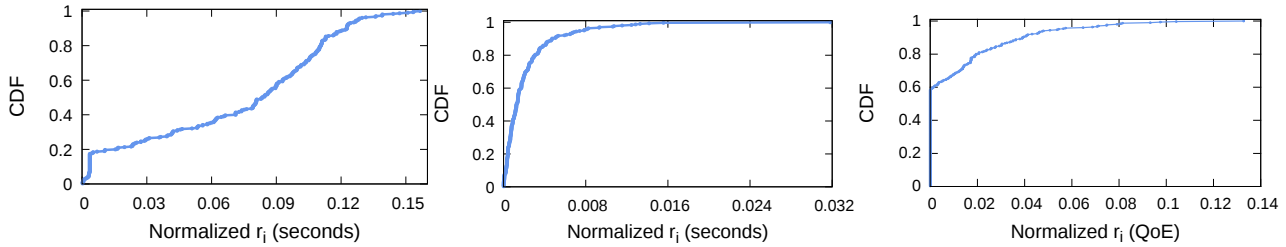
(a) Quality (average bitrate) - Zone 1 (b) Quality (average bitrate) - Zone 2 (c) Quality (average bitrate) - Zone 3

Figure 10: Residual error CDFs for quality (average bitrate)



(a) Quality switch - Zone 1 (b) Quality switch - Zone 2 (c) Quality switch - Zone 3

Figure 11: Residual error CDFs for the quality switch



(a) Cumulative Stall time (b) Startup delay (c) QoE

Figure 12: Residual error CDFs for stall time, startup delay and QoE estimation

presented in Table 3. Based on the results shown by Mao et al. [16], the q function was set to linear, where q is equal to the bitrate. In addition (also according to [16]), the stall and startup weights (μ and ω) were set to 4.3. The value of the quality switches constant (λ) was tuned to 1 [32]. Finally, the zones weights (α_1, α_2 and α_3) were empirically set to 0.7, 0.3 and 0, for Zone 1, Zone 2 and Zone 3. The reason behind setting α_3 to zero comes from the perfect prediction scenario considered in the evaluation. In such cases, the FoV will correspond 100% of tiles of Zones 1 and 2. Thus, there is no influence of the quality of Zone 3 on the user' s perception. In the case that perfect prediction would not be possible, the weights would need to be tuned accordingly.

The performance of the method is assessed by means of the residual error calculated between real data sample (entry in the training set) and the predicted one (as already introduced in Section 5.1 and Equation 3). With the purpose of generalizing the method for videos of arbitrary duration, the residual error for the metrics average bitrate, quality switch and startup delay are normalized by the factor N of the residual error equation, which corresponds to the

Table 3: Constants and function values assigned to the function to estimate QoE (refer to Equation 2)

Param.	Value
q	Linear
μ	4.3
ω	4.3
λ	1
α_1	0.7
α_2	0.3
α_3	0

considered video length (200 seconds). Figures 10, 11 and 12 show the Cumulative Distribution (CDF) of the residual error for the four VR playout performance metrics and the QoE estimation.

Looking at the quality prediction capacities of PERCEIVE (Figures 10(a) to 10(c)), it is possible to observe that the residual errors are very small (224 Kb/s and 220 Kb/s for Zones 1 and 2, respectively, for over 90% of the cases). If normalized by the maximum available quality (6.0 Mb/s), it represents only 3.73% and 3.67% of residual error. This means that in roughly 97% of the cases, the quality levels are correctly predicted. Even though the residual error for Zone 3 is slightly higher (4.5%), it is still within the acceptability range.

The accuracy of the quality switch prediction (Figures 11(a) to 11(c)) shows even better results. For over 90% of the samples, Zones 1, 2 and 3 present a residual error of $r_{i_1} \leq 0.00745$, $r_{i_2} \leq 0.01604$ and $r_{i_3} \leq 0.01877$, respectively. In line with the findings for the average bitrate prediction, Zone 3 presented a higher residual error (1.9%), as this is the zone with the highest number of quality switches during the video playout. In Figure 11(b) it is possible to observe that on over 80% of samples the residual error is zero. This is because the quality switch behavior for both extreme cases of the network performance is predictable: first, when the network performance is sufficiently high, the rate adaptation will stabilize at the highest representation, and no further quality switches are expected. Second, when the network performance is degraded, the rate adaptation will keep the video playout at the lowest available quality representation, and, similarly, no further switches are expected.

The stalling time (Figure 12(a)) shows an error close to 13% for over 90% of the testing samples. One main reason behind such increased residual error is the wide range of the predicted variable (as we saw in the regression tree of Figure 9(a)). Nevertheless, several samples in the training dataset presented zero seconds of stall time. We found that such predictable cases are associated with high levels of network performance. For each of these samples, a residual error of 0.95 was accounted (as 0.95 is the lowest predicted value). As the presented regression tree is the optimal prune, further growth would lead to overfitting, and thus a higher cross-validation error. Due to the relatively high stall time for intermediate and degraded network performance, the prediction performance is impaired as the network performance degrades. However, at high levels of stall time, the QoE is already completely degraded. Thus, the increased error does not impair the accuracy of the QoE estimation.

The final VR playout parameter, the startup delay (Figure 12(b)), is characterized as the elapsed time between the request of the video and the playout of the first segment. In the considered context, the startup delay prediction presented a well predictable pattern with $r_i \leq 0.00473$ for over 90% of the cases. Also, the regression tree presented a stable prediction performance across all the evaluated samples.

Finally, Figure 12(c) depicts the residual error for the QoE estimation. By applying the QoE model defined in Section 4.2 to each sample i, it is possible to estimate QoE for both the predicted playout values and the original ones. Then, the residual error can be calculated. Through this procedure, the QoE estimation error induced by the proposed prediction scheme can be assessed. As shown in Figure 12(c), the QoE estimation presents $r_i \leq 0.03922$ for over 90% of the cases.

6 CONCLUSION

Virtual Reality applications based on adaptive tile-based video streaming are booming, as VR content becomes available to the general public. To be able to cope with their ultra-high bandwidth and low latency requirements, network and services providers are required to assess the end-client perceived performance of such services.

In this paper we presented PERCEIVE, a novel VR performance evaluation method to assess the user's perception of the VR content when streamed through the network. By means of machine learning techniques applied to the network performance indicators, it predicts the adaptive VR performance both in terms of VR main playout parameters (quality, quality switches, stalling time and starting time) and the perceived QoE. To our knowledge, this is the first VR performance model.

PERCEIVE has been evaluated considering a real-world environment, in which VR videos are streamed while subjected to an LTE/4G network performance. Then, we assessed its accuracy by means of the residual error between the predicted and measured values. PERCEIVE is able to predict the playout performance metrics with an average prediction error lower than 3.7% and, the perceived quality with a prediction error lower than 4% for over 90% of all the tested cases. PERCEIVE not only provides very high prediction accuracy, but also allows analyzing the influence of networks on the VR streaming parameters. This feature has helped us pinpoint the network delay as the QoS feature that affects the transport of VR services the most.

We believe our work is one step forward in the assessment of VR applications performance, which is an open subject in the state-of-the-art on multimedia network management. Although the proposed QoE model has not been validated through subjective tests, we believe it is an acceptable approach considering the scope of this work. As we are evaluating the predictability of the QoE indicator based on the performance of the application layer, possible adjustments in the weights of Equation 1 will not affect the prediction error of the QoE indicator. Aiming at providing realistic weights for Equation 1, as future work, we intend to perform subjective tests of the proposed QoE model. We also intend to explore and improve the estimation capabilities of our approach, focusing on viewport prediction and on adaptive streaming heuristics.

ACKNOWLEDGEMENT

This research was performed partially within the project G025615N "Optimized source coding for multiple terminals in self-organizing networks" from the fund for Scientific Research-Flanders (FWO-V). This work was also partially funded by CAPES, CNPq, FAPERGS and IFSul PROPESP.

REFERENCES

[1] Arslan Ahmad, Alessandro Floris, and Luigi Atzori. 2016. QoE-centric service delivery: A collaborative approach among OTTs and ISPs. *Computer Networks* 110 (2016), 168 – 179. https://doi.org/10.1016/j.comnet.2016.09.022

[2] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. 1161–1170. https://doi.org/10.1109/BigData.2016.7840720

[3] BBC. 2017. Facebook: We want a billion people in VR. (2017). https://goo.gl/2LNuAo Accessed 16-October-2017.

[4] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. 2009. TCP Congestion Control. RFC 5681. (Sept. 2009). https://doi.org/10.17487/RFC5681
[5] Cisco. 2017. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021.* Technical Report. Cisco Systems.
[6] C. Concolato, J. Le Feuvre, F. Denoual, E. Nassor, N. Ouedraogo, and J. Taquet. 2017. Adaptive Streaming of HEVC Tiled Videos using MPEG-DASH. *IEEE Transactions on Circuits and Systems for Video Technology* PP, 99 (2017), 1–1. https://doi.org/10.1109/TCSVT.2017.2688491
[7] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2017.7996611
[8] R. I. T. da Costa Filho, W. Lautenschlager, N. Kagami, V. Roesler, and L. P. Gaspary. 2016. Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE. In *2016 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOCOM.2016.7842022
[9] E. Dahlman, G. Mildh, S. Parkvall, J. Peisa, J. Sachs, Y. Selén, and J. Sköld. 2014. 5G wireless access: requirements and realization. *IEEE Communications Magazine* 52, 12 (December 2014), 42–47. https://doi.org/10.1109/MCOM.2014.6979985
[10] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 513–526. https://doi.org/10.1145/2987443.2987459
[11] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360 Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM, New York, NY, USA, 67–72. https://doi.org/10.1145/3083165.3083180
[12] M. Hosseini and V. Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. 107–110. https://doi.org/10.1109/ISM.2016.0028
[13] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. 2016. CFA: A Practical Prediction System for Video QoE Optimization. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, Santa Clara, CA, 137–150. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/jiang
[14] Xianshu Jin, Hwiyun Ju, Sungchol Cho, Boyeong Mun, Cheongbin Kim, and Sunyoung Han. 2016. QoS routing design for adaptive streaming in Software Defined Network. In *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. 1–6. https://doi.org/10.1109/ISPACS.2016.7824694
[15] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Stanford, CA, 1137–1145.
[16] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843
[17] Sreerama K. Murthy. 1998. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Min. Knowl. Discov.* 2, 4 (Dec. 1998), 345–389. https://doi.org/10.1023/A:1009744630224
[18] H. Nam, K. H. Kim, and H. Schulzrinne. 2016. QoE matters more than QoS: Why people stop watching cat videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524426
[19] Omar A. Niamut, Emmanuel Thomas, Lucia D'Acunto, Cyril Concolato, Franck Denoual, and Seong Yong Lim. 2016. MPEG DASH SRD: Spatial Relationship Description. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 5, 8 pages. https://doi.org/10.1145/2910017.2910606
[20] Pradrip Paudyal, Federica Battisti, and Marco Carli. [n. d.]. Impact of video content and transmission impairments on quality of experience. *Multimedia Tools and Applications* 2016 ([n. d.]). https://doi.org/10.1007/s11042-015-3214-0
[21] PERCEIVE. 2018. Performance Estimation for VR Videos. (2018). https://github.com/rtcostaf/PERCEIVE Accessed 2-April-2018.
[22] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages. https://doi.org/10.1145/2818361
[23] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*. ACM, New York, NY, USA, 306–314. https://doi.org/10.1145/3123266.3123453
[24] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 Video Delivery over Cellular Networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC '16)*. ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/2980055.2980056
[25] R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
[26] Muhammad Zubair Shafiq, Jeffrey Erman, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia Wang. 2014. Understanding the Impact of Network Dynamics on Mobile Video User Engagement. *SIGMETRICS Perform. Eval. Rev.* 42, 1 (June 2014), 367–379. https://doi.org/10.1145/2637364.2591975
[27] Iraj Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia* 18, 4 (2011), 62–67.
[28] M. Torres Vega, C. Perra, and A. Liotta. 2018. Resilience of Video Streaming Services to Network Impairments. *IEEE Transactions on Broadcasting* (2018). https://doi.org/10.1109/TBC.2017.2781125
[29] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.
[30] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, and T. D. Hämäläinen. 2015. Kvazaar HEVC encoder for efficient intra coding. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1662–1665. https://doi.org/10.1109/ISCAS.2015.7168970
[31] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 193–198. https://doi.org/10.1145/3083187.3083210
[32] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 325–338. https://doi.org/10.1145/2829988.2787486

APPENDIX

This appendix provides a detailed description of the procedure to be followed in order to allow reproducibility of the experiments performed in this work. All the employed datasets and source code are available at the PERCEIVE repository [21].

In Section 1.1 we present an overview of the experimental design and its general setup. Next, the re-encoding procedure required to generate tile-based VR-videos is explained in Section 1.2 Then, in Section 1.3, the VR video player considered in this work is examined. Subsequently, both the network and the VR video playout datasets are thoroughly discussed in Section 1.4. Finally, in Section 1.5, we give an overview of the R scripts responsible for performing the machine learning task.

1.1 Experimental Procedure Overview and Specifications

The experimental procedure is split into three steps, as explained in Section 5.1 (Figure 4). First, the VR video player requests tile-based videos, from a web server, while subjected to controlled network conditions. The VR video player is responsible for measuring and recording the VR video playout performance, while the network conditions are enforced by Linux Traffic Control (TC) mechanism. Second, the VR video playout performance indicators are given as input to the machine learning process. At this stage, machine learning is responsible for characterizing how each network condition impacts the video playout performance. Finally, in the third step, an estimation of QoE is provided by giving the VR video performance as input to the QoE model.

To perform the first step, we employ three dedicated virtual machines deployed on the imec iLab.t Virtual Wall emulation platform[5]. The first machine was used to run the VR video player, while the second was used to host tile-based VR videos using a regular Apache web server. Through traditional IP routing and Linux Traffic Control (TC), the third machine was configured as a gateway between the other two, acting as a network condition enforcement point. Each virtual machine was configured with a quad-core Intel

[5]imec iLab.t: http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

Xeon E3-1220 v3 CPU running at 3.10GHz, 15GB RAM, 16GB of storage and running Linux Ubuntu 14.04 (3.13.0-33). The full list of packages and its respective versions is available at PERCEIVE's repository [21] (Setup/packages.txt).

Steps two and three do not require any specific hardware or software specification. Step two was performed using R (1.0.143) [25], and for the third step we employed a simple electronic spreadsheet to compute the QoE model (Section 4.2) over the VR video playout performance indicators. After this overview, the remainder of this section will cover practical details of the main elements of the experiment.

1.2 Tile-based HAS VR-video Re-encoding

In order to generate tile-based HAS VR-videos, it was necessary to re-encode the original VR videos from Wu et al.' s dataset [31] (namely "Google Spotlight-HELP" and "Freestyle Skiing"). Herein, the re-encoding procedure is explained step-by-step.
After downloading the original VR-videos "Google Spotlight-HELP"[6] and "Freestyle Skiing"[7], the raw videos must be first extracted using the following command of FFMPEG[8]:

```
$ ffmpeg -i inVideo.mkv -c:v rawvideo
outVideo.yuv
```

Next, the HEVC tile-based version of the videos is generated using Kvazaar[9]. Kvazaar splits the videos based on the generated YUV file, the desired tiling scheme, resolution and frames per second (FPS), as shown in the following example. This command is to be executed per video quality.

```
$ kvazaar -i outVideo.yuv --input-res
3840x2160 -o outVideo12x4.hevc --tiles
12x4 --slices tiles --mv-constraint
frametilemargin -q 30 --period 30
--input-fps 30
```

Subsequently, each of the tiles of the VR-video is packed into an mp4 container employing the MP4Box software [10].

```
$ MP4Box -add outVideo12x4.hevc:split_tiles
-fps 30 -new video_tiled_4K_12x4.mp4
```

Finally, based on the desired length of the HAS segment, the per-tile per segment files of the VR-video are extracted. For example, the following command defines one second for the segment length, 12x4 tiling scheme and three video resolutions (720p, 1080p and 4K). This procedure also generates MPD files by using multiple quality representations.

```
$ MP4Box -dash 1000 -rap -frag-rap
-profile live -out has_tiled_12x4.mpd
../SOURCE/video_tiled_720_12x4.mp4
../SOURCE/video_tiled_1080_12x4.mp4
../SOURCE/video_tiled_4K_12x4.mp4
```

[6]https://youtu.be/G-XZhKqQAHU
[7]https://youtu.be/0wC3x_bnnps
[8]FFMPEG: https://www.ffmpeg.org/
[9]Kvazaar: https://github.com/ultravideo/kvazaar
[10]MP4box: https://gpac.wp.imt.fr/mp4box/

1.3 VR Video Player

Both the source code and binary for the VR video player are available at the PERCEIVE repository [21] (VR-player/Source and VR-player/bin respectively). The player provides support to variable tiling scheme and can be adapted to several QoE zone schemes (Section 3.1). Additionally, the player supports viewport traces (a previously recorded log regarding the user's head track) as input. The player is written in C language and employs Curl library to perform HTTP requests. The player also allows parameters to be passed through command line arguments. It is particularly useful when running large experiments, so that the player parameterization can be done dynamically by an external script. The full set of parameters is shown in Table 5. For example, the following player call is used for requesting the first 60 segments of the video named "video2", available at the IP "10.0.0.251", using the viewport trace stored in the file "user1/video2.txt", using 100 seconds timeout and a 12x4 tiling scheme. In this case, the resultant VR-video playout performance will be written in the file named "video2playout".

```
$ VR-player 10.0.0.251 video2 60
video2playout user1/video2.txt 100 4 12
```

Table 4: VR-video player command line arguments.

Sequence	Description
1	IP address of the video server
2	Video filename
3	Number of segments to download
4	Output filename (to write playout performance results)
5	Viewport trace filename (head track logs)
6	Session timeout (max number of seconds)
7	Number of vertical tiles (tiling scheme)
8	Number of horizontal tiles (tiling scheme)

1.4 Video Playout and Network Datasets

The file "Sample.csv" (directory "Network dataset" [21]) provides the 48 network conditions considered in our experiments. The conditions were extracted from [29] and adapted according the procedure described in Section 5.2. The range for each input parameter is summarized in Table 1. The configuration ID is the leftmost field in the file "Sample.csv", followed by the fields throughput TCP (Mb/s), delay (msec) and packet loss rate (%). After parsed, these values are given as input to the Linux TC, which act as a network condition enforcement point.

In turn, the file "playoutPerformance.txt" (directory "Playout performance dataset" [21]) provides the resultant output of the first step of the experimental procedure (described in Section 1.1). Furthermore, this is the same file given as input to the machine learning process (step two). Along with the network dataset, Table 5 summarizes the input parameters for generating the playout performance dataset.

Table 6 shows the set of fields of the resultant VR-video playout performance. Fields 1 - 5 are related to the video input parameters, listed in Table 5. Fields 6 - 8 are related to the network conditions. Fields 9 - 22 corresponds to the VR-video playout performance measured by the VR-video player. Finally, fields 23 - 25 are calculated as the number of per-zone tiles times the average bitrate for each video resolution. In the PERCEIVE repository, we provide a

Table 5: PERCEIVE video input parameters.

Parameter	Value/Range	Details
VR video	Google Spotlight-HELP and Freestyle Skiing	V1 and V2 (from [31])
Head track traces	Google Spotlight-HELP and Freestyle Skiing	V1 and V2 (from [31])
Video format	MP4 - HEVC tile-based and HAS	Using MP4Box
Video encoder	Kvazaar	Kvazaar encoder [30]
HAS	720p (1.8Mb/s), 1080p (2.7Mb/s) and 4K (6Mb/s)	Kvazaar encoder [30]
Segment size	1 second	From [11]
Tiling scheme	8x4 and 12x4	From [24]
Considered viewport	One central tile and eight border tiles	Section 3.1

bash script "addQuality.sh" (directory Scripts) which can be used to perform this computation.

Table 6: Fields sequence of the file "playoutPerformance.txt" (Field 1 is the leftmost value in the file).

Field	Description	Type/unit
1	Video ID	string
2	User ID	string
3	Tile format	horizontal X vertical
4	Network trace ID	string
5	Experiment round	integer
6	TCP throughput	Mb/s
7	Delay	msec
8	Packet loss	%
9	Number of tiles 720p for Zone 1	integer
10	Number of tiles 1080p for Zone 1	integer
11	Number of tiles 4K for Zone 1	integer
12	Number of tiles 720p for Zone 2	integer
13	Number of tiles 1080p for Zone 2	integer
14	Number of tiles 4K for Zone 2	integer
15	Number of tiles 720p for Zone 3	integer
16	Number of tiles 1080p for Zone 3	integer
17	Number of tiles 4K for Zone 3	integer
18	Number of quality switches for Zone 1	integer
19	Number of quality switches for Zone 2	integer
20	Number of quality switches for Zone 3	integer
21	Stall time	seconds
22	Startup delay	seconds
23	Average bitrate for Zone 1	Mb/s
24	Average bitrate for Zone 2	Mb/s
25	Average bitrate for Zone 3	Mb/s

1.5 Machine Learning

The directory "R Scripts" [21] provides all the source code used to generate the regression decision trees shown in Section 5.3. Each of the eight decision trees has its own source code (R script). In addition to the R tool [25], we employed the following packages: stargazer [11], gdata [12], rpart [13], tree [14] and rpart.plot [15]. Finally, it is worth mentioning that the trees shown in this work were obtained through their optimal prune. Which means that during the prune stage, we selected the complexity parameter (CP) associated with the minimum cross-validation error (xerror).

[11] https://cran.r-project.org/web/packages/stargazer/index.html
[12] https://cran.r-project.org/web/packages/gdata/index.html
[13] https://cran.r-project.org/web/packages/rpart/index.html
[14] https://cran.r-project.org/web/packages/tree/index.html
[15] https://cran.r-project.org/web/packages/rpart.plot/index.html

APPENDIX E — PAPER SUBMITTED TO THE ACM TOMM

- **Title**: *Dissecting the Performance of VR Video Streaming Throughthe VR-EXP Experimentation Platform*
- **Journal**: ACM Transactions on Multimedia Computing, Communications, and Applications
- **Qualis**: B1
- **Notes**: Submitted on 17/Feb 2019

Abstract. To cope with the massive bandwidth demands of Virtual Reality (VR) video streaming, both the scientific community and the industry have been proposing optimization techniques such as viewport-aware streaming and tile-based adaptive bitrate heuristics. As most of the VR video traffic is expected to be delivered through mobile networks, a major problem arises: both the network performance and VR video optimization techniques have the potential to influence the video playout performance and the Quality of Experience (QoE). However, the interplay between them is neither trivial nor has it been properly investigated. To bridge this gap, in this paper we introduce VR-EXP, an open-source platform for carrying out VR video streaming performance evaluation. Furthermore, we consolidate a set of relevant VR video streaming techniques and evaluate them under variable network conditions, contributing to an in-depth understanding of what to expect when different combinations are employed. To the best of our knowledge, this is the first work to propose a systematic approach, accompanied by a software toolkit, which allows one to compare different optimization techniques under the same circumstances. Extensive evaluations carried out using realistic datasets demonstrate that VR-EXP is instrumental in providing valuable insights regarding the interplay between network performance and VR video streaming optimization techniques.

Dissecting the Performance of VR Video Streaming Through the VR-EXP Experimentation Platform

ROBERTO IRAJA TAVARES DA COSTA FILHO, Institute of Informatics - UFRGS, Brazil
MARCELO CAGGIANI LUIZELLI, Federal University of Pampa, Brazil
STEFANO PETRANGELI, Adobe Research, USA
MARIA TORRES VEGA, JEROEN VAN DER HOOFT, TIM WAUTERS, FILIP DE TURCK, Ghent University - imec, Belgium
LUCIANO PASCHOAL GASPARY, Institute of Informatics - UFRGS, Brazil

12

To cope with the massive bandwidth demands of Virtual Reality (VR) video streaming, both the scientific community and the industry have been proposing optimization techniques such as viewport-aware streaming and tile-based adaptive bitrate heuristics. As most of the VR video traffic is expected to be delivered through mobile networks, a major problem arises: both the network performance and VR video optimization techniques have the potential to influence the video playout performance and the Quality of Experience (QoE). However, the interplay between them is neither trivial nor has it been properly investigated. To bridge this gap, in this paper we introduce VR-EXP, an open-source platform for carrying out VR video streaming performance evaluation. Furthermore, we consolidate a set of relevant VR video streaming techniques and evaluate them under variable network conditions, contributing to an in-depth understanding of what to expect when different combinations are employed. To the best of our knowledge, this is the first work to propose a systematic approach, accompanied by a software toolkit, which allows one to compare different optimization techniques under the same circumstances. Extensive evaluations carried out using realistic datasets demonstrate that VR-EXP is instrumental in providing valuable insights regarding the interplay between network performance and VR video streaming optimization techniques.

CCS Concepts: • **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Virtual reality**; • **Networks** → *Network protocols*; *Public Internet*.

Additional Key Words and Phrases: Virtual reality, adaptive streaming, quality of experience, quality of service

ACM Reference Format:
Roberto Iraja Tavares da Costa Filho, Marcelo Caggiani Luizelli, Stefano Petrangeli, Maria Torres Vega, Jeroen van der Hooft, Tim Wauters, Filip De Turck, and Luciano Paschoal Gaspary. 2019. Dissecting the Performance of VR Video Streaming Through the VR-EXP Experimentation Platform. *ACM Trans. Multimedia Comput. Commun. Appl.* 1, 1, Article 12 (January 2019), 23 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Virtual Reality (VR) video streaming applications are becoming increasingly popular. VR Head-Mounted Displays (HMDs) are expected to grow from 18 million in 2017 to nearly 100 million by 2022, while the associated network traffic is expected to increase 12-fold [5]. The same study points out VR video streaming as a key application, which has the potential to significantly increase the VR penetration. VR video streaming applications are challenging due to three main reasons: (*i*) they will run mostly over mobile networks, as mobile devices will account for 71% of the total

1551-6857/2019/1-ART12 $15.00
https://doi.org/0000001.0000001

IP Internet traffic by 2022 [5]; *ii*) mobile networks are characterized by highly variable levels of performance [7]; and (*iii*) VR video streaming applications demand high levels of network performance to achieve a satisfactory Quality of Experience (QoE) [5]. To provide a notion of how demanding these applications are, recent studies have shown that, to provide adequate levels of QoE, current VR video applications require a network delay lower than 9 ms [8], while the bandwidth needs for the upcoming ultra high definition VR will reach 500 Mbps [5]. At this level of demand, not only will network operators struggle to provide cost-effective services, but VR video content providers and developers will also be challenged by such resource-intensive applications.

To overcome the aforementioned challenges, both the academy and industry are investigating novel approaches for improving the efficiency of the VR video streaming ecosystem. In this direction, efficient spherical-to-plane projection schemes, which include tile-based VR video streaming, are prominent strategies for reducing the bandwidth requirements imposed by VR videos [4, 6, 12, 14, 16, 38]. These investigations extend well-established approaches to 2D video streaming, such as HTTP Adaptive Streaming (HAS) and Dynamic Adaptive Streaming over HTTP MPEG-DASH paradigms [9, 27]. In the first step, VR videos are encoded at different quality levels and representations (*e.g.*, 720p, 1080p, 4K, 8K). Subsequently, they are split into both spatial tiles and temporal segments. During the streaming session, the client will only request tiles corresponding to its viewport (*i.e.*, the visible portion of the full 360-degree panoramic view). To perform selective tile requests, these schemes rely on viewport prediction heuristics [11, 15, 17, 22, 25, 26]. Other crucial building blocks in this ecosystem are Adaptive Bitrate (ABR) streaming and Buffer Management heuristics. ABR streaming benefits from both the temporal/spatial segmentation and predicted viewport to manage the playout buffer. To do so, it requests for each segment the tiles that are estimated to belong to the viewport in high quality, while the remaining tiles will either be requested at lower quality variants or not fetched at all [2, 12, 13, 19, 25].

Optimization schemes, such as the ones just mentioned, contribute to minimizing the use of network resources (mainly in terms of bandwidth). For example, when prioritizing high-resolution representations only for tiles in the viewport, a bandwidth reduction of up to 72% can be achieved [14]. However, optimization approaches can impair the performance of the VR video streaming severely, thus, degrading the user's perception of the service (*i.e.*, QoE). For example, consider the case where the predicted viewport tiles are downloaded in advance. Errors in the viewport prediction for the user's field of view will lead to QoE degradation even though the bandwidth is high enough for the application requirements. Furthermore, the video encoding and streaming decisions (such as the spherical-to-plane projection strategy, tiling scheme, available quality representations, frame rate) and the client-side implementation aspects (*e.g.*, playout buffer size, rate adaptation heuristics, and tile fetching method) play an essential role in shaping the resulting VR video playout performance and, ultimately, QoE. Finally, it is worth noting that these parameters and heuristics may perform quite differently when subjected to variable network performance conditions. We deem that distinct groups can benefit from a solution to this problem: (*i*) both the research community and VR solutions designers can carry out far-reaching evaluation of their approaches when subjected to complex and realistic scenarios; and (*ii*) considering that network operators already have tools in place for measuring network performance, they can estimate VR video application performance and QoE experienced by their subscribers.

When considering both the multitude of approaches to optimize a VR video streaming and the highly variable mobile network performance, it becomes a difficult challenge to understand how different (combinations of) optimization techniques perform under varying infrastructure conditions. The lack of a publicly-available method and tools for systematic and reproducible evaluation exacerbate this challenge. To fill in this gap, in this paper, we propose VR-EXP, an adaptive VR video streaming experimentation platform. The platform is capable of systematically evaluating different

combinations of VR video streaming optimization approaches. Also, VR-EXP allows pinpointing the interplay between a set of optimization techniques and variable network performance. Comprised of an evaluation method and software components, VR-EXP assumes as input tile-based VR videos, network datasets, and parameters (e.g., network performance conditions, users' head-tracking information, ABR heuristics, and tile fetching methods). Then, it emulates essential components of the VR video streaming ecosystem, measuring key VR video playout performance indicators. Finally, our platform produces, as output, detailed VR video playout performance and QoE estimation reports. Using VR-EXP, we carry out an in-depth analysis of (combinations of) state-of-the-art VR video optimization approaches under varying network conditions.

We summarize the contributions of this work as follows:

- We provide a platform to carry out systematic evaluations that can be executed across realistic scenarios.
- Throughout an extensive evaluation, we provide an in-depth analysis of the performance of cutting-edge optimization approaches for VR video streaming.

The remainder of this paper is organized as follows. In Section 2, we present an overview of background concepts and state-of-the-art optimization approaches for the VR video ecosystem. In Section 3, we introduce VR-EXP, encompassing its main components and design choices. In Section 4, we outline the evaluation setup including the considered parameters and datasets. Then, in Section 5 we present and discuss the main results. Finally, our conclusions along with perspectives for future work are presented in Section 6.

2 BACKGROUND AND STATE OF THE ART

In this section, we provide a thorough description of state-of-the-art optimization techniques for VR video streaming. We organize these investigations into three research groups. We start by reviewing relevant projection schemes for VR video encoding. Next, we evaluate prominent investigations regarding viewport prediction. Finally, we evaluate adaptive bitrate streaming and buffer management approaches for VR videos.

2.1 Spherical-to-Plane Projection

One effective strategy to reduce the huge bandwidth demands of 360-videos is delivering only the viewport in high resolution, streaming the remaining area of the video in low resolution or not at all. To achieve this spatial segmentation of the panoramic view, several approaches explore spherical-to-plane projection techniques [4, 6, 12, 14, 16, 38]. For example, Graf *et al.* [12] examine the bitrate overhead and bandwidth requirements of distinct tiling schemes (*i.e.*, 1x1, 3x2, 5x3, 6x4 and 8x5) implemented using modern video codecs (*e.g.*, HEVC/H.265 and VP9). By applying Peak Signal-to-Noise Ratio (PSNR) within the VR video viewport, the authors assess the video quality and conclude that the 6x4 tiling scheme provides the best trade-off among viewport selection flexibility, bitrate overhead, and bandwidth requirements. In a similar direction, Zhou *et al.* [38] further examine this field by comparing standard spherical projection approaches to offset projection techniques. The latter are characterized by distorting the spherical surface to allow the convergence of the pixels of the VR video in a particular direction. Offset projections are significantly more complex than traditional projection techniques because they demand a simultaneous control of bitrate and view orientation adaptations. By employing PSNR and Structural Similarity (SSIM), the authors conclude that, in general, offset projections can provide better quality than their non-offset counterparts. Despite their contributions, the conclusions of these investigations are limited because they do not consider important variables, such as the effects of variable viewport prediction error and

parallel fetching methods (such as HTTP/2) on their approaches. Also, the mentioned approaches are evaluated considering limited network performance conditions.

In another important investigation, Chen *et al.* [4] analyze recent advancements regarding alternative projection methods, including viewport-dependent and viewport-independent approaches. The central objective of this work is to assess both the coding efficiency and distortion introduced by each approach. Besides valuable quantitative and qualitative insights regarding a wide range of projection schemes, the authors conclude that in order to effectively evaluate such a wide range of projection schemes, a more sophisticated evaluation process is required. The main reason for this conclusion is that traditional PSNR computes the whole projection map, which cannot handle viewport-dependent projections. Additionally, due to the unpredictability of viewport prediction errors, the areas surrounding the viewport should also be considered in the quality evaluation, but with a reduced weight. In this investigation, the authors also review alternative metrics for video quality assessment proposed by JVET [3]. They conclude that although several flaws of conventional PSNR have been fixed, a more comprehensive method for evaluating video quality for viewport-dependent VR videos is still missing.

2.2 Viewport Prediction Algorithms

Viewport prediction heuristics benefit from the tile-based structures of the VR video to enable differentiated handling of group of tiles. Since a full VR video can easily reach 12K video resolution [6], most video players rely on heuristic algorithms to predict near-future user's head movements. Considering the next position prediction, the VR video emulator is able to keep a small playout buffer (*e.g.*, 2 seconds) requesting only tiles that are likely to belong to the viewport, which ultimately leads to reduced bandwidth utilization. In this direction, several recent investigations propose viewport prediction algorithms [11, 15, 17, 22, 25, 26].

To illustrate how the viewport prediction works, consider the example of a user watching a tile-based VR video using a head-mounted display. Assume a given temporal segment S_k and a respective viewport V_k, as depicted in Figure 1 (a). At this moment, the video player is requesting high-resolution chunks only for tiles inside the viewport V_k. Then, based on the viewport prediction for the next segment (S_{k+1}), the video player starts requesting high-resolution tiles for the predicted viewport V_{k+1} (delimited by the blue dashed square in Figure 1 (b)). However, rather than moving his/her head up, consider that the viewer actually slightly moves to the right (see Figure 1 (c)). At this point, due to the viewport predictor error, the VR player requested seven tiles in high-resolution which will not actually be displayed (upper left red tiles in Figure 1 (d)). Likewise, seven low-resolution tiles end up belonging to the viewport (bottom right red tiles in Figure 1 (d)). As one can observe, viewport prediction is a sensitive task. Viewport prediction errors may lead to partial or full degradation of the perceived quality, even if the network performance conditions are enough to guarantee the user's QoE.

To perform the viewport prediction, most approaches follow a similar procedure, which includes processing one or more input information, applying a prediction method, and then checking the prediction accuracy. As input, prediction algorithms can rely on past users' head motion [15, 24, 26], fixation point acceleration [22], fixation point angular velocity [11, 22, 25], image saliency maps and motion maps [11], or even sound localization information [17]. In turn, to perform the viewport prediction itself, state-of-the-art approaches rely on deep learning [15], mathematical modeling [17, 22, 24–26], or neural networks [11]. Finally, the prediction accuracy is assessed by subjecting the prediction model to traces containing realistic head-tracking information (*i.e.*, ground truth). Thus, the residual error can be evaluated. By performing such predictions, the VR video player can, according to He *et al.* [14], reduce bandwidth utilization in up to 72%.

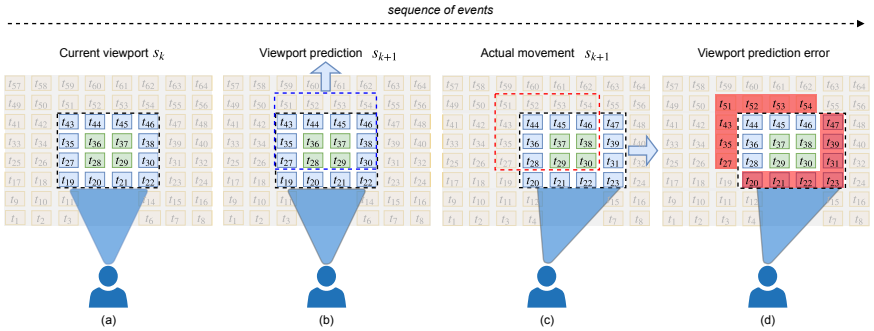


Fig. 1. Working principles of the viewport prediction and the viewport error.

As discussed, although prediction algorithms may present acceptable accuracy under certain circumstances, prediction errors are very likely to occur due to the randomness of users behavior. Besides, prediction algorithms may considerably decrease their accuracy when the size of the playout buffer is increased. For example, the prediction accuracy can drop from 90% to approximately 60% if the prediction window is increased from 1 to 2 seconds [26]. However, an increased playout buffer may be crucial to operate in current mobile networks, which are characterized by highly variable performance conditions, even in short time frames. Considering these intricacies, an effective assessment of viewport prediction algorithms should consider (and quantify) how the error rate of a particular algorithm affects QoE when combined with other optimizations (*e.g.*, buffer management heuristics and dynamic rate adaptation algorithm) and subjected to realistic network performance.

2.3 Adaptive Bitrate Algorithms and Buffer Management

Taking viewport prediction information as input, most approaches rely on per tile rate adaptation algorithms. This method allows reducing the amount of information to be downloaded by keeping only the viewport's tiles in high resolution. However, this task is far from trivial, even for traditional video streaming. ABR algorithms are complex because they must manage the available bandwidth while maximizing the quality representation and minimizing the stall probability. Although ABR algorithms for traditional 2D video streaming have been extensively explored, recent investigations [1, 28] show that it is still an open research problem. As an example, it is demonstrated the possibility to significantly improve the performance of state-of-the-art ABR algorithms, namely BOLA [29] and MPC [36]. Akhtar *et al.* [1] demonstrate that both BOLA and MPC algorithms rely on parameters that are sensitive to variable network performance, so they may perform poorly under certain conditions. To fill this gap, the authors introduce VirtualPlayer [1], a trace-based simulator that mimics the behavior of a traditional video streaming player. It allows, for example, to investigate ABR algorithms when subjected to real-world networks. In the same direction, Spiteri *et al.* [28] introduce Sabre, an open-source simulation tool that enables simulating ABR algorithms for 2D videos when subjected to realistic requirements.

When it comes to VR video, ABR becomes a much more challenging task. State-of-the-art approaches for adaptive bitrate in VR videos differ from each other mainly with respect to how they manage the balance between video quality and available bandwidth while considering the spatial segmentation. For example, Petrangeli *et al.* [25] consider a multi-zone VR video and propose a per tile quality selection heuristic. The algorithm starts by selecting the highest available quality for the inner tiles (close to the fixation point), and then repeats this procedure for the outer zones until the residual bandwidth is exhausted. This approach alleviates the edge effect (transition between different quality representations). Thus, it provides superior VR video quality at the cost of increased bandwidth consumption.

He *et al.* [13] propose to simultaneously optimize, among other parameters, playout bitrate and buffer occupancy. Similarly to Petrangeli *et al.* [25], they perform bitrate adaptations depending on the position of each tile concerning the current fixation point. However, they introduce a learning strategy with the ability to avoid performance degradation for future segments by automatically adapting the buffer reservation. By using a fine-grained bitrate adaptation, these investigations were able to reduce the bandwidth utilization in 35% and 40%, respectively.

Graf *et al.* [12] advance a step forward in the state of the art by providing a comprehensive investigation with respect to essential components of the VR ecosystem. The authors introduce three tile scheme strategies for ABR, namely Full Delivery Basic, Full Delivery Advanced and Partial Delivery. These schemes drive the ABR algorithm regarding the bitrate adaptation for both the viewport and the remaining tiles. For example, in Full Delivery Basic scheme, all the tiles belonging to the viewport are requested in the highest available quality, while the remaining tiles are requested in the lowest quality, regardless of the available bandwidth. The Partial Delivery scheme employs an aggressive bandwidth saving strategy, requesting only the tiles within the viewport in high resolution, while the remaining tiles are not requested at all. The authors evaluate several projection schemes (as discussed in Subsection 2.1), combined with multiple segment sizes. By assessing the bitrate overhead, bandwidth requirements, and viewport quality, this approach achieve bandwidth savings from 40% to up to 65% when compared to state-of-the-art techniques.

Closely related to ABR algorithms, the playout buffer management plays a vital role in the VR video realm. As discussed earlier, an increased playout buffer size consists in an effective way to protect against stalls (*i.e.*, empty buffer) caused by network performance fluctuations. On the other hand, a small playout buffer is necessary to keep the accuracy of viewport prediction methods within acceptable levels. Specifically on this subject, Ma *et al.* [19] propose a dynamic buffer size management method which is guided by a constrained optimization model. This method aims at maximizing QoE by adjusting the buffer size based on the viewport prediction error and available bandwidth. Throughout simulation experiments, the authors claim gains from 2.7% up to 6.7%, in terms of QoE, when compared to non-dynamic buffer size approaches. In another relevant investigation, Almquist *et al.* [2] present a data-driven study which explores the trade-off between the playout buffer size (*i.e.*, prefetching aggressiveness) and viewport prediction errors. The prefetching aggressiveness is evaluated while considering different VR video categories (*i.e.*, exploration, static, moving, rides and misc.). The authors provide valuable qualitative and quantitative insights regarding how to best address the prefetching aggressiveness trade-off. As a key insight, they demonstrate that the accuracy of the prediction varies significantly among different categories. Additionally, in line with previous investigations, they emphasize that adequate levels of viewport prediction accuracy are observed only within a very small time frame.

In summary, during a streaming session, several components (*e.g.*, projection scheme, viewport prediction error, buffer management approach, dynamic rate adaptation algorithm, and network performance) will have a major influence on the user's QoE. Despite several research efforts, little is known about the interplay between a set of VR video components and variable network performance conditions. Also, a solution to provide an in-depth and reproducible evaluation of the VR video streaming ecosystem is still missing. In this paper, we introduce VR-EXP, an open-source publicly-available platform for evaluating adaptive VR video streaming performance and QoE when subjected both to multiple VR video optimization techniques and variable network performance conditions. Different from the related work, instead of evaluating each optimization technique independently (or within a reduced set), our approach provides an extensible VR video emulator that allows for the simultaneous evaluation of multiple state-of-the-art optimization techniques. Combined with the provided network controller and realistic network performance dataset, VR-EXP contributes a step forward to the VR field by providing a reproducible method for evaluating

adaptive VR video streaming optimization approaches. To the best of our knowledge, this is the first open-source method and toolkit for a comprehensive evaluation of the VR video ecosystem.

3 METHODOLOGY

In this section, we introduce the VR-EXP platform. We start by presenting the general scheme, highlighting its inputs, outputs, and main modules. Then, we introduce the VR video client emulator and its main components. Next, we discuss alternatives for enforcing network performance conditions. Finally, we present the considered QoE model, which allows evaluating the effects of multiple VR video optimization techniques on QoE.

3.1 VR-EXP General Scheme

In a nutshell, the VR-EXP platform enables evaluating the interplay between a set of adaptive tile-based VR streaming optimizations and variable network performance conditions. Figure 2 depicts the main modules of VR-EXP. The proposed method consists of systematically fetching VR videos through a controlled network environment. From a client perspective, the adaptive VR video client emulator coordinates the use of several VR video techniques upon requesting VR videos from a content server. During the streaming session, the Network Performance Enforcement Point enforces realistic network conditions on the network links between the VR Video Client Emulator and the Content Server. Once the VR video streaming session is finished, VR-EXP reports key VR video playout performance metrics.

The most important module is the VR video client emulator, which is responsible for processing the input parameters, emulating state-of-the-art VR video optimization approaches, and measuring the playout performance. Except for the rendering process, the VR video client emulator mimics the behavior of a VR video adaptive streaming client. Although the rendering task is important for the VR video context, it is mostly related to the HMD rendering capabilities of each device. Therefore, in this work we are interested in evaluating the influence of variable network performance on VR adaptive streaming in an isolated manner, without the interference of HMD particularities. To emulate a dynamic network topology as well as enforcing real-world conditions, VR-EXP relies on either an SDN network controller or the Linux Traffic Controller. In the proposed platform, adaptive VR videos are delivered by an HTTP server (Apache[1]), which delivers tile-based VR videos in multiple quality representations according to the HAS scheme.

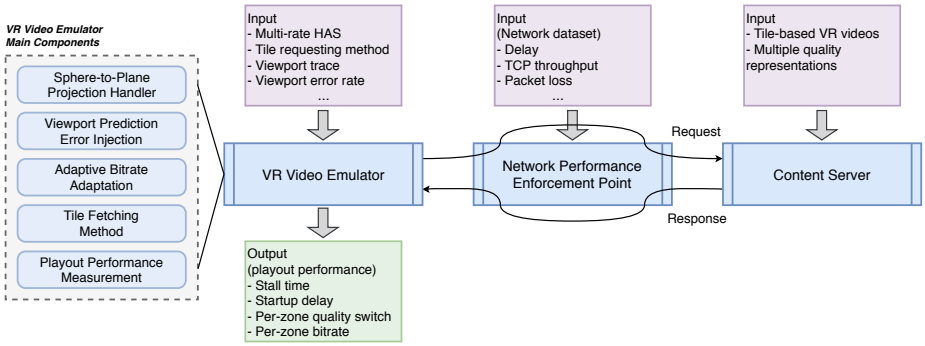


Fig. 2. VR-EXP general scheme.

The emulation of the entire VR video streaming ecosystem requires the configuration of several parameters and inputs. For flexibility, VR-EXP enables the definition of its parameters at run time.

[1]Apache HTTP Server: https://httpd.apache.org/

It allows building scripts for automating complex and extensive experiments. For example, it is possible to parameterize the VR video client emulator by defining behavior characteristics such as the tile requesting method, the rate adaptation heuristic, the expected viewport prediction error, and so forth. In turn, the network module is expected to be fed with a dataset containing a set of network performance metrics (*e.g.*, delay, packet loss rate, TCP throughput). It then enforces these conditions into the emulated links connecting the VR video client emulator to the content server. Once all the input datasets and parameters are configured, the VR video client emulator starts fetching VR videos using the HTTP protocol. After processing the VR video, the emulator writes an output file containing the processed VR video performance metrics, as well as the raw performance data, as described in Section 3.2. The complete set of source code and datasets related to the VR-EXP platform are released under GNU General Public License v3.0 and are publicly-available at GitHub[2].

3.2 VR Video Client Emulator

We now focus on the high-level overview of the main functional components of the VR video client emulator[3]. The VR-EXP video client emulator is an extensible and fully parameterized headless VR video client emulator. The source code is written in the C language using the Curl[4] library to systematically fetch tile-based VR videos over the HTTP protocol. The emulator is composed of five main components (Figure 2): (*i*) sphere-to-plane projection handling, (*ii*) viewport prediction error injection, (*iii*) adaptive bitrate adaptation, (*iv*) tile fetching method, and (*v*) playout performance measurement. Next, we describe their functionality.

Sphere-to-Plane Projection Handling. Several state-of-the-art approaches for VR video streaming optimization rely on tile-based projection schemes [8, 12, 15, 25]. Additionally, modern QoE estimation models employ tile clustering methods for manipulating groups of tiles in a coordinated way, depending on their spatial position [8]. To cope with these features, VR-EXP is designed to support different tiling schemes and tile clusterization into multiple zones. A multi-zone approach is in line with the notion that the spatial position in which a VR video degradation occurs is vital for estimating QoE. For example, Figure 3 depicts an 8x5 tiling scheme which is divided into three zones, where Zone 1 is defined as containing only the viewport's central tile, Zone 2 encompassing the viewport border tiles (8 tiles), and Zone 3 containing the 31 remaining tiles.

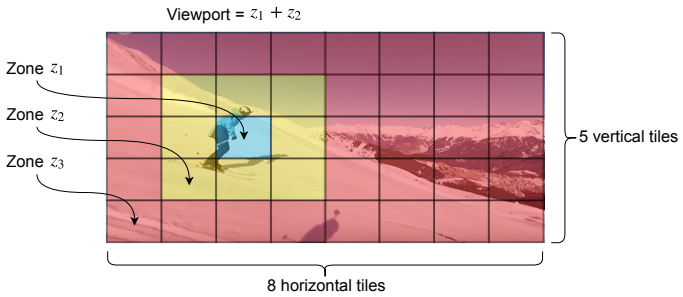


Fig. 3. Example of an 8x5 tiling scheme organized in three zones.

Viewport Error Injection. Once the projection handler is capable of dealing with several tiling schemes, the next step towards efficient VR streaming consists of emulating the viewport prediction. More precisely, in order to provide an accurate simulation of the entire VR context, the most

[2]VR-EXP: https://github.com/rtcostaf/TOMM2019_VR-EXP/
[3]For additional details, please refer to the documentation available at VR-EXP [34].
[4]Curl: https://curl.haxx.se/libcurl/c/

significative information regarding any heuristic is the viewport prediction error. As discussed in Subsection 2.2, viewport prediction algorithms present highly variable accuracy depending on many factors. To allow for an accurate evaluation of error patterns, VR-EXP provides a controlled viewport error injection during the streaming session. We designed a flexible viewport error injection component which takes as input viewport traces (*i.e.*, datasets describing the coordinates where the users have looked at in a particular time frame). The viewport trace files contain a full record of coordinates of the VR video, captured at regular intervals (*e.g.*, at each 20ms). In order to provide a mechanism to evaluate the impact of wrong viewport predictions, VR-EXP enables injecting artificial prediction errors when processing the coordinates specified in the trace file. The error injection mechanism can be parameterized with a given error rate, as well as easily extended to support different error models. Using the viewport error injection can be very helpful in designing novel viewport prediction algorithms. Using VR-EXP, the developer can indeed measure the impact of the error on the resulting performance of the VR sessions. Also, VR-EXP allows understanding what would be the minimum error to guarantee a target performance. The error injection mechanism can be parameterized with a given error rate, as well as easily extended to support different error models. In the current version of VR-EXP, we modeled the viewport prediction error as a random variable with a uniform distribution.

Dynamic Bitrate Adaptation. Taking advantage of the viewport prediction, ABR algorithms provide significative bandwidth savings by selecting appropriate quality representations for each spatial zone. In this procedure, each of the zones is assigned with the most suitable quality level according to both their distance from the center of the viewport and the available bandwidth. VR-EXP currently implements two alternative adaptive streaming heuristics. The general idea of the first heuristic procedure, named Full Delivery (FD) [25], is as follows. Once knowing the available bandwidth in the network (*i.e.*, based on network conditions measured during the download of previous segments), the emulator downloads tiles with the best fit regarding the available bandwidth. For each segment, the heuristic tries to first increase the bitrates on the inner zones of the viewport (Zone Z1 in Figure 3). Then, it repeats the procedure to stream tiles from the outer zones (Zones 2 and 3, respectively). Thus, when considering networks with enough available bandwidth, this heuristic will increase the quality representation for all zones. This approach provides effective protection against viewport prediction errors, at the cost of high bandwidth consumption. The second heuristic, named Full Delivery Basic (FDB) [11, 26], works similarly to the first one. However, instead of increasing the bitrate whenever possible in outer zones, this heuristic increases the bitrates only for zones within the viewport. Although FDB reduces the amount of consumed bandwidth significantly, it may entail QoE degradation in case of viewport prediction errors. Regardless of the approach, the downloaded segments are stored in a playout buffer to be eventually played. Observe that the buffer size plays a significant role in the VR video client performance – particularly regarding viewport prediction accuracy – and, therefore, can be adjusted as needed.

Tile Request Method. On the one hand, the combined use of tile-based VR videos, ABR heuristics, and viewport prediction have proven to be an effective approach to avoid wasting bandwidth. On the other hand, the adaptive tile-based video encoding leads to an increased number of files to be fetched from the content server. For example, consider a 10-minute tile-based VR video, split into 1-second segments, encoded with an 8x5 tiling scheme, and available on three quality representations (*i.e.*, HD, FHD and 4K). For each second of video, it would be necessary to download one quality representation for each tile, which leads to 40 files per second, that is, 24,000 files for a 10-minute streaming session. Considering the above, for each video segment, there is a set of tiles within pre-specified zones to be fetched from the server. VR-EXP allows fetching VR tiles according to two strategies: serial and parallel. On using the serial request method, tiles are fetched

from the server, one by one, using multiple (non-parallel) HTTP requests, in a single connection. In turn, in the parallel method, tiles within the same zone (*e.g.,* tiles belonging to the viewport) are fetched in parallel using a configurable number of parallel connections. VR-EXP allows specifying the number of simultaneous connections per zone and splits the set of tiles uniformly among the available connections. It worth mentioning that VR-EXP employs a regular HTTP server (*e.g.*, Apache, NGINX) for hosting the tile-based VR videos, so no specific parameterization is required.

VR Video Playout Performance. To bring together the components detailed throughout this section, along with realistic input datasets, VR-EXP provides a realistic emulation of the VR video streaming ecosystem. Therefore, the next important step toward building a comprehensive VR video evaluation platform is to measure the VR video playout performance accurately. During the video streaming session, VR-EXP assesses a number of VR video playout performance metrics capable of objectively characterizing the quality of the video playout. These metrics include the number of tiles per zone/quality (*e.g.*, number of tiles within the viewport retrieved in 4K resolution), number of quality switches per zone (*i.e.,* number of quality switches on a specific zone), stall time and startup time delay. Table 1 provides an example list regarding the playout performance metrics provided by VR-EXP (the complete list may vary depending on the parameterization of the VR-EXP). These metrics were selected because they are the most influential when predicting QoE based on the video streaming playout performance [8]. It is worth mentioning that VR video applications rely on TCP/HTTP for providing reliable streaming services. Thus, network performance degradation events, such as packet loss or increased delay, will necessarily translate into either or both quality switches and video stall. Along these lines, VR-EXP focuses on evaluating how multiple VR video optimization techniques interact with variable network performance conditions. Evaluating the distortion introduced by different projection schemes and codecs is out of the scope of this work.

Table 1. Playout performance metrics for VR video streaming

Field	Short description	Type/unit
1	Video file name	string
2	User ID (viewport trace ID)	integer
3	Tiling scheme (8x4, 12x4)	string
4	Network trace ID	integer
5	Tile request method (serial, mutli-thread)	integer
6	Network TCP throughput - downlink	Mbps
7	Network delay - RTT	ms
8	Network packet loss - downlink	%
9	Number of 720p tiles - zone 1	integer
10	Number of 1080p tiles - zone 1	integer
11	Number of 4K tiles - zone 1	integer
12	Number of 720p tiles - zone 2	integer
13	Number of 1080p tiles - zone 2	integer
14	Number of 4K tiles - zone 2	integer
15	Number of 720 tiles - zone 3	integer
16	Number of 1080p tiles - zone 3	integer
17	Number of 4K tiles - zone 3	integer
18	Number of quality switches - zone 1	integer
19	Number of quality switches - zone 2	integer
20	Number of quality switches - zone 3	integer
21	Total video stall time	seconds
22	Video startup delay	seconds
23	Average bitrate - zone 1	Mbps
24	Average bitrate - zone 1	Mbps
25	Average bitrate - zone 1	Mbps
26	Viewport error rate (0 - 100)	%

3.3 Network Performance Enforcing

In order to enforce real-world network performance conditions, it is possible to employ, at least, three different strategies: *(i)* network simulation, *(ii)* network emulation or *(iii)* dedicated network infrastructure. The use of network simulation provides great control over the simulated elements. However, simulating the full VR video components stack, plus complex network aspects (*e.g.*, routing, fairness between distinct TCP flavors, operating system features and their limitations) would burden the complexity of implementation and potentially lead to inaccurate simulation results. On the other extreme, dedicated infrastructure provides a realistic environment at the cost of reduced flexibility and complex setup. In light of this, we decided to employ network emulation as we consider this design choice a suitable balance between flexibility and accuracy.

For emulating network links, VR-EXP provides a customized SDN controller (based on Ryu[5]) which, along with Mininet[6], enables reproducing sophisticated network scenarios. The SDN controller is the preferred option for complex network environments due to its ability to easily handle dynamic network topologies and forwarding rules. Also, this strategy allows evaluating the VR video streaming ecosystem when subjected to large topologies and high link competition through many concurrent video sessions. However, if the network scenario does not require such complexity (*e.g.*, simulating a few links with static routes), the SDN approach could be replaced with a simpler alternative mechanism (*i.e.*, Traffic Control[7]). Both approaches can benefit from simplified scripting to read input datasets, which describe the network performance (*i.e.*, delay, jitter, residual bandwidth, packet loss) and enforce these network conditions on a target network.

3.4 QoE Model

VR-EXP is designed to work with any QoE model that supports VR video playout performance indicators as input. Employing a QoE model can be very insightful as it provides a consolidated view regarding the effect of multiple VR video playout performance metrics on QoE. In consonance with state-of-the-art QoE models for traditional video streaming [20, 23, 37], we employ a QoE model [8] that is able to translate multiple VR video playout performance characteristics into an estimated QoE score.

$$\phi(Z_k) = \overbrace{\sum_{\forall t \in Z_k} \sum_{\forall c \in C(t)} q(R(C_{t_m}))}^{\text{Quality}} - \overbrace{\mu \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left(\frac{d_c(R_c)}{C_c} - B_c \right)_+}^{\text{Stalls}} - \underbrace{\lambda \cdot \sum_{\forall t \in Z_k} \sum_{\forall c \in C_{t_m}} \left| q(R(C_{t_{m+1}})) - q(R(C_{t_m})) \right|}_{\text{Quality switches}} - \underbrace{\omega \cdot T_s}_{\text{Startup}} \tag{1}$$

The QoE model is comprised of four main terms, as shown in Equation 1. The first term uses a function q which translates the measured bitrate of the chunk tm (function R) into the quality perceived by the user. Function q is in line with the notion that different users may have a different perception regarding the bitrate of the VR video. For example, some users may have a linear perception, which means that an increase of 50% in the video bitrate will be perceived as an increase

[5]Ryu SDN Controller: https://osrg.github.io/ryu/
[6]Mininet: https://mininet.org
[7]TC: http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html

of 50% in quality. In turn, other users may have a sub-linear quality perception, where the same increment in terms of bitrate is perceived as a marginal increment of quality. The second term is used to keep track of the stall time. It considers, for each chunk c, that a stall event occurs when the download time $\frac{d_c(R_c)}{C_c}$ is higher than the playout buffer (B_c) when the chunk download started. Therefore, the total stall time is given by $\sum_{c=1}^{C}\left(\frac{d_c(R_c)}{C_c} - B_c\right)_+$. In addition, $|q(R_{ct+1}) - q(R_{ct})|$ considers the quality switches between consecutive chunks and T_s tracks the startup delay. Finally, constants μ, λ, ω are the non-negative weights used to adapt the model to different user sensitivities regarding degradation in VR video playout. For example, a higher value of μ with respect to the other weights means that the user is more susceptive to video stalls. Consequently, these events should affect the QoE indicator more severely.

Aiming to provide a more realistic assessment, the considered QoE model resorts to the concept of zones. The main idea of this approach relies on the notion that the QoE estimation must consider the spatial segmentation aspect of the VR videos. For example, tiles near to the center of the viewport will greatly steer the quality perceived by the user, while bad qualities on tiles of the edge zones, or even outside the viewport, will potentially go unnoticed. For this reason, the overall video QoE ($\phi(V)$) is modeled as a weighted linear sum of the QoE measurement per zone (Equation 2). Each weight ($\alpha_1, \alpha_2, ..., \alpha_k$) determines the relative importance of each zone.

$$\phi(V) = \alpha_1 \cdot \phi(Z_1) + \alpha_2 \cdot \phi(Z_2) + ... + \alpha_k \cdot \phi(Z_k) \tag{2}$$

4 EVALUATION SETUP

Using VR-EXP as basis, we carry out an extensive evaluation of state-of-the-art heuristics when subjected to variable network performance conditions. In this section, we present the experimental setup. We start by introducing the 4G/LTE performance dataset, which provides realistic network conditions to the evaluation process. Next, we describe the VR video dataset, including head track traces, which enables the evaluation of viewport-aware approaches. We end this section by outlining the experiment plan and its main procedures.

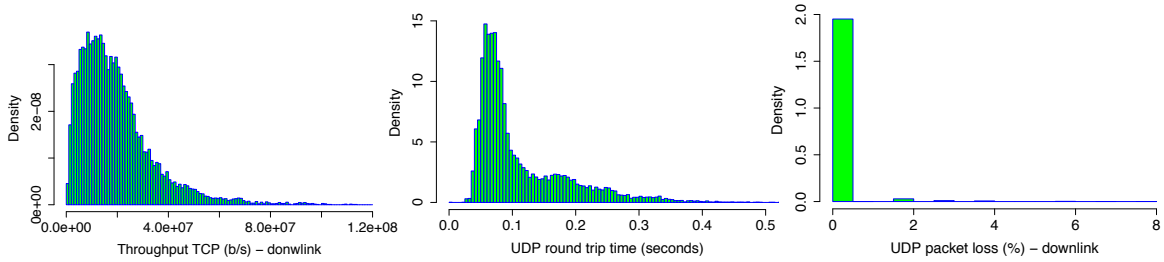
4.1 4G/LTE Performance Dataset

In this work, along with the VR-EXP method and toolkit, we provide a comprehensive dataset for 4G/LTE network performance. The dataset contains the following IP metrics: Round Trip Time (RTT), delay variation (also referred to as jitter), one-way packet loss, and one-way TCP throughput (in the scope of this work also referred as to residual bandwidth). These metrics were gathered by means of IP active measurements, in conformance with the recommendations issued by the IETF IP Performance Metrics Working Group [21]. To obtain these indicators we employed a scalable active measurement-based platform named Netmetric [7, 10, 30]. Table 2 describes the dataset fields. Except for packet loss, all values are averaged over the whole packet burst.

Table 2. IP performance dataset (Field 1 is the leftmost column

Field	Short description	Type/unit
1	TCP throughput - uplink	b/s
2	TCP throughput - downlink	b/s
3	Round-trip time (RTT)	seconds
4	Packet loss - downlink	%
5	Packet loss - uplink	%
6	Delay variation (jitter) - uplink	seconds
7	Delay variation (jitter) - downlink	seconds

In Figure 4 we present a brief statistical analysis of the measurements available in the dataset regarding the three main metrics. As shown in Figure 4a, the TCP throughput metric presents a wide range of measured values for the downlink. For example, the downlink presents a throughput varying from a minimum of 31.4 Kbps to a maximum of 113.2 Mbps, with a median of 16.5 Mbps and a mean of 19.6 Mbps. In turn, Figure 4b depicts the RTT metric ranging from 1 ms up to 18.5 seconds (the upper limit is not shown in the Figure 4b due to the long tail), with a median of 81 ms and a mean of 120 ms. Finally, the packet loss (Figure 4c) for the downlink ranges from 0% up to 8%.



(a) TCP throughput - downlink (b) Delay - RTT (c) Packet loss - downlink

Fig. 4. Network performance dataset: histograms for TCP throughput, delay and packet loss.

The network dataset comprises over 14,000 measurements taken from 01/06/2017 to 31/07/2017. Each measurement considers the end-to-end path between the source node, a server located at the premises of the Federal University of Rio Grande do Sul, and a measurement device (destination). The destination of each measurement session is an Android (6.0) smartphone running the measurement agent and attached to a 4G/LTE network. Measurement devices were spread countrywide embracing the four major mobile operators. Together, these operators are responsible for providing mobile services to over 236 million subscribers [31].

Each measurement session is composed of two bi-directional packet bursts, where the first uses UDP and the second TCP. The UDP packet burst is employed to measure RTT, loss and jitter by injecting 400 packets of 100 bytes at 50 ms intervals. As some operators block the Network Time Protocol (NTP), we decided not to measure the One-Way Delay (OWD). Instead, the RTT metric was obtained based on a single clock (source). In turn, the TCP burst gauges the TCP throughput for the considered path by injecting 640 packets of 1,488 bytes each. For privacy reasons, sensitive information regarding the considered mobile operators (*e.g.*, operator name, provider ID, cell ID) has been removed from the dataset.

Considering the number of measurements and the wide range of the considered metrics, the network performance dataset may be useful to support further research in several areas. Especially in the field of VR video streaming since the available metrics encompass the network performance indicators that influence video streaming performance the most (*i.e.*, delay and residual bandwidth) [8, 37]. Additionally, the metrics' ranges allow evaluating high-resolution and tile-based VR videos, including 4K+ resolution. It is worth mentioning that the range for the TCP throughput metric is in line with similar studies conducted in other regions *et al.* [32].

4.2 VR Video Dataset

In this evaluation we use two VR videos from Wu et al.' s dataset [35], namely "Google Spotlight-HELP" and "Freestyle Skiing". Aiming at evaluating viewport-aware approaches, for each video we also consider the available datasets which describe users' head movements while watching the VR videos. However, the original VR videos are non tile-based, so they needed to be re-encoded. To

do so, the first step consisted of extracting the raw YUV files, making use of the Kvazaar encoder [33]. The resulting encoding produced two tiling schemes: 8×4 and 12×4 [18, 26]. Additionally, each tiling scheme was encoded into three quality representations, namely 720p (1.8Mbps), 1080p (2.7Mbps) and 4K (6Mbps). Next, we employed the MP4Box[8] application to pack the encoded videos into MP4 containers. Then, we sliced each quality representation into 1 second segments. Finally, we used MP4Box to extract per-tile files and to generate the MPEG Dash Media Presentation Description (MPD) files considering multiple quality representations. Table 3 summarizes the main parameters regarding the VR video dataset.

Table 3. Adaptive streaming configurations.

Videos	Qualities (bitrates)	Quality zones	Segment	Tiling
Google Spotlight Freestyle Skiing (Wu et al. [35])	720p - 1.8Mbps 1080p - 2.7Mbps 4K - 6Mbps	Zone 1: 1 tile (central FoV) Zone 2: 8 tiles (adj. Zone 1) Zone 3: remaining tiles	1 s	12×8 8×4

4.3 Experiment Plan

VR-EXP was deployed on the imec iLab.t Virtual Wall emulation platform[9]. The experiments consisted of employing VR-EXP for measuring VR video performance while subjected to a broad variety of network conditions and multiple VR video optimization techniques. To capture the interplay between the considered variables (detailed in Subsection 3.2), we varied the experiment's parameters (*e.g.*, network performance, VR video, tiling scheme, adaptive bitrate heuristic, playout buffer size) in a controlled manner. The experiments were organized around each key VR video optimization technique, namely the viewport prediction error, per tile rate adaptation heuristics and tile requesting method. In a first step, we varied the parameters within each heuristic at a time, assuming default values for the remaining heuristics (according to Table 4). In order to capture the interplay within a set of heuristics, in the second step, we carried out a more sophisticated evaluation by varying multiple parameters and heuristics within the same experiment. To instantiate the QoE model, we consider the three-zone scheme defined by Da Costa Filho *et.al* [8], where Zone 1 refers to the viewport center tile, Zone 2 encompasses the eight tiles surrounding Zone 1, and Zone 3 includes all remaining tiles. We also consider the same constants and function values proposed by the authors, which are summarized as follows (refer to Equations 1 and 2): $q = Linear$, $\mu = 4.3$, $\omega = 4.3$, $\lambda = 1$, $\alpha_1 = 0.7$, $\alpha_2 = 0.3$, and $\alpha_3 = 0$.

5 RESULTS

In this section, we present the results regarding the application of VR-EXP along with the inputs and parameters described in Section 4. We start by evaluating the effects of the Viewport Prediction Error (VPE) on VR video playout performance and QoE. Next, we extend this analysis to encompass per tile rate adaptation heuristics, and finally to tile requesting method. We end this section by presenting a more sophisticated scenario, where multiple parameters, heuristics and the network performance conditions vary within the same experiment.

5.1 Effects of Viewport Prediction Error

When dealing with traditional 2D video streaming, we use the term video bitrate (*e.g.*, 2 Mbps, 6 Mbps) equivalently with their respective representations of quality (*e.g.*, 1080p, 4K). Also, we can

[8]MP4Box https://gpac.wp.imt.fr/mp4box/
[9]imec iLab.t: http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

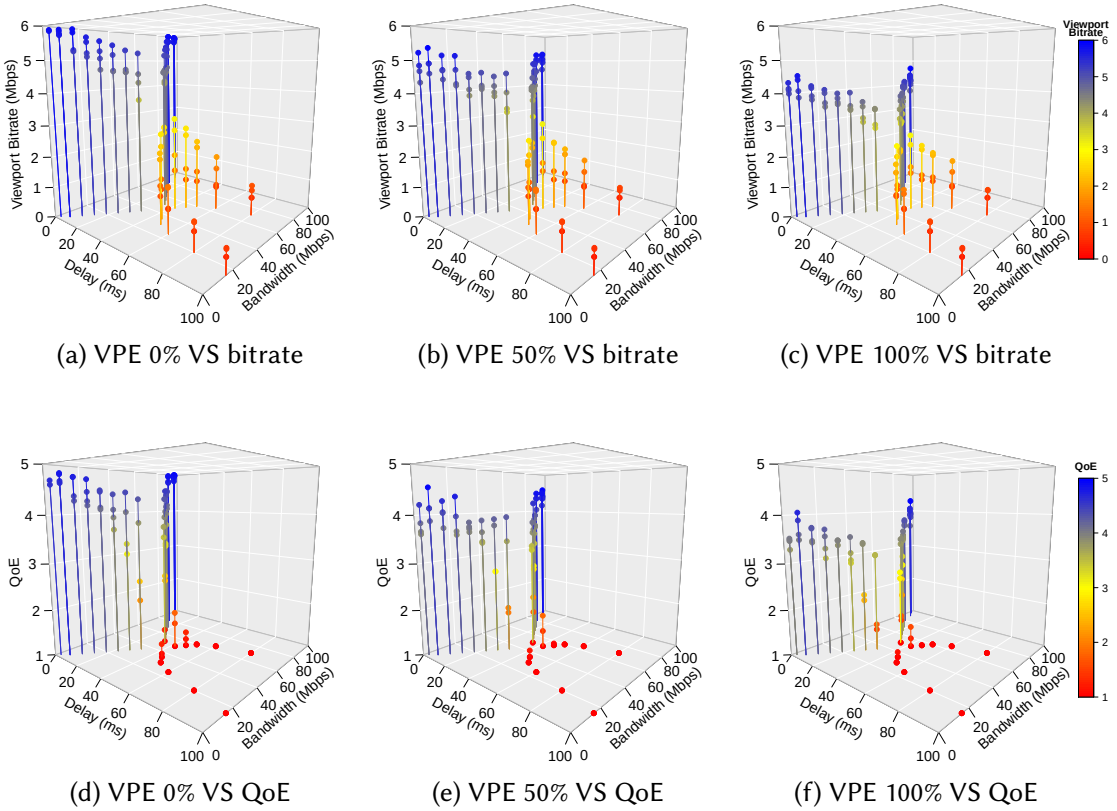
Table 4. Main VR-EXP input parameters

Parameter	Value/Range	Details
VR video	Google Spotlight-HELP and Freestyle Skiing	Both videos are used in all experiments
Head track traces	Google Spotlight-HELP and Freestyle Skiing	multiple users/head track traces for each video
Video format	MP4 - HEVC tile-based and HAS	Using MP4Box[10]
Video encoder	Kvazaar	Kvazaar encoder [33]
HAS	720p (1.8Mbps), 1080p (2.7Mbps) and 4K (6Mbps)	Kvazaar encoder [33]
Segment size	1 second	the same for all experiments
Tiling scheme	8x4 and 12x4	Both tiling schemes are used in all experiments
Considered viewport	One central tile and eight border tiles	NA
Viewport error rate	0% up to 100%	Default 0%
Rate adaptation heuristic	FD and BFD	Default BFD
Tile request method	Single thread, 6 threads and 8 threads	Default single thread
Playout buffer	2 sec up to 8 sec	Default 2 sec

state that there is a correspondence between the average bitrate delivered to the user and the average bitrate that effectively traversed the network (*i.e.*, bandwidth consumption). However, when it comes to tile-based VR video streaming, this relationship becomes less trivial. For example, consider the streaming of a tile-based VR video using a 12x4 tiling scheme and a viewport containing nine tiles. Assume that during most of the streaming session the viewport is displayed in 4K resolution, while the tiles outside the viewport are fetched at 720p. It turns out that the bitrate delivered to the user (visible portion of the VR video) is equivalent to the 4K representation (*i.e.*, 6 Mbps). However, when considering the FDB heuristic for adaptive bitrate, the overall bitrate of the video (*i.e.*, equivalent to the average bandwidth demand during the streaming session) will be slightly higher than the bitrate of the 720p representation. It happens because most of the video (not visible by the user) was fetched in low resolution. For didactic reasons, in this evaluation we use the term *Viewport Bitrate* to denote the bitrate *perceived* by the user, while the term *Video Bitrate* refers to the total bitrate of the video (averaged over all tiles), being equivalent to the bitrate effectively demanded from the network.

As discussed in Section 2, depending on the viewport prediction algorithm and the playout buffer size, the viewport prediction accuracy can be quite erratic. In this section, we apply VR-EXP to evaluate the impact of the viewport prediction errors on both video playback performance and QoE. Figure 5 shows the performance of the video playout, regarding viewport bitrate and QoE, when subjected to variable network performance conditions and prediction error. Figure 5a illustrates the baseline scenario, characterized by absence of viewport prediction errors. In this scenario, a network delay below 12 ms is fundamental to provide good levels of viewport bitrate (recall that the bitrate for the 4K representation is 6 Mbps). In such conditions, it is possible to observe viewport rates close to 6 Mbps across a wide range of available bandwidth values.

Figures 5b and 5c show how the viewport prediction error affects the viewport average bitrate. When considering a viewport prediction error rate equal to 50% (Figure 5b), the maximum bitrate decreases approximately by 1 Mbps, while a 100% error in the viewport (Figure 5c) drops the maximum bitrate to near 4 Mbps, even when considering the most favorable network condition. The viewport error does not affect the playout performance when subjected to significantly degraded levels of network performance (*i.e.*, delay higher than 50 ms). In such cases, the rate adaptation algorithm has no room for increasing the quality representation. All tiles are requested at the lowest available quality representation and, as a direct consequence, a viewport error does not lead to additional degradation. Figures 5d, 5e and 5f demonstrate the impact of prediction errors on QoE. One can observe that severe prediction errors (Figure 5f) may lead to a decrease of up to 2 points in the QoE score when compared to the baseline scenario shown in Figure 5d.



(a) VPE 0% VS bitrate (b) VPE 50% VS bitrate (c) VPE 100% VS bitrate

(d) VPE 0% VS QoE (e) VPE 50% VS QoE (f) VPE 100% VS QoE

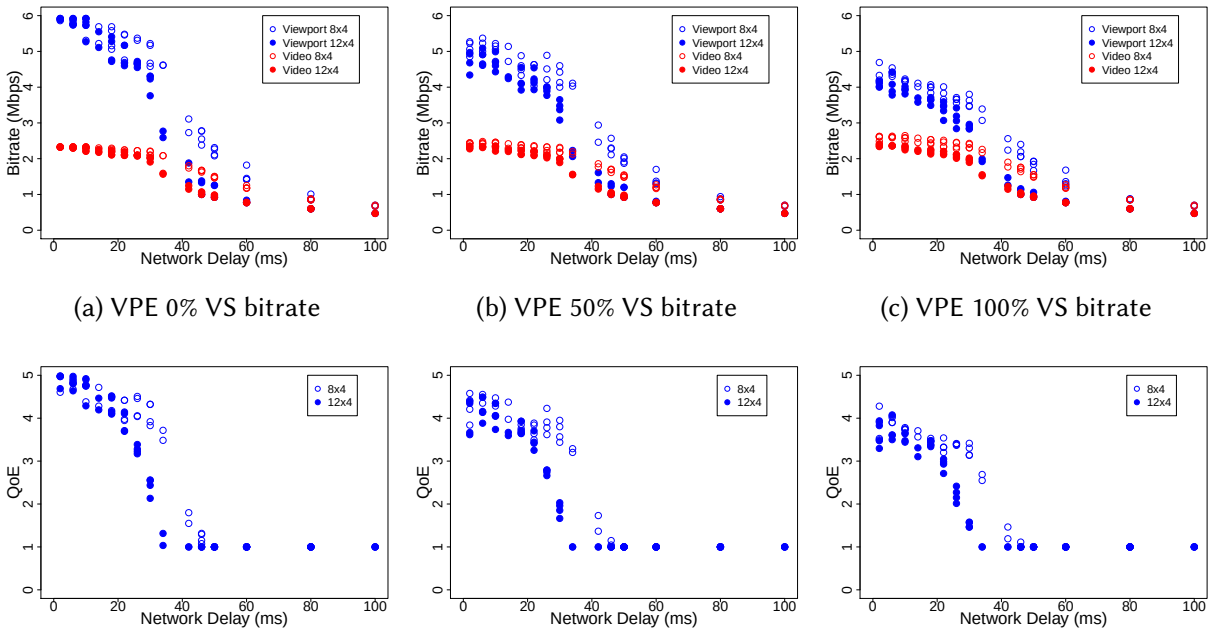
Fig. 5. The effects of the viewport prediction error on VR video playout performance and QoE.

Next, we employed VR-EXP to assess more accurately the effects of the viewport prediction error. To do so we added the tile scheme information. Moreover, we split the rates between the bitrate observed for the tiles within the viewport and the bitrate for the entire video (including the viewport). Figure 6a shows the baseline case, which considers a perfect viewport prediction. To improve readability, in all plots of Figure 5 we show the network variability only in terms of delay, removing the bandwidth dimension from the analysis. The red dots represent the bitrate for the entire VR video (*i.e.*, viewport + remaining tiles), which is equal to the network bandwidth required for streaming the VR video. When it comes to the viewport (blue dots), both tiling schemes are able to achieve the maximum bitrate when the delay is lower than 12 ms. However, the 8x4 tiling scheme presents significantly better bitrates for intermediate network conditions (delay between 12 ms and 60 ms). This gain is explained by the fact that the HTTP request/response overhead is lower for the 8x4 tiling scheme (32 files per segment) against 48 files per segment for the 12x4 tiling scheme. When the delay is higher than 60 ms, the video playout is totally impaired, and neither the tiling scheme nor the VPE introduces additional degradation.

Complementing the previous analysis, in Figures 6b and 6c it is possible to observe that the tiling scheme plays an important role in the video playout performance. The viewport error leads to lower viewport bitrate for intermediate network conditions when compared to the baseline scenario. Still, for intermediate network delay, the 8x4 tiling scheme presents a viewport bitrate up to 2 Mbps higher when compared to the 12x4 tiling scheme. The obtained results indicate that the VPE influences, mainly, the average viewport bitrate and quality switch metrics. The remaining metrics for playout performance (*i.e.*, startup delay and stall time) are not affected by prediction

errors. Figures 5d, 5e and 5f show that, in line with previous findings, the viewport prediction error has the potential to reduce the QoE score significantly. Nevertheless, the tiling scheme can dramatically influence the QoE score. For example, in Figure 6d it is possible to observe that, for a network delay of around 35 ms, the 8x4 tiling scheme outperforms the 12x4 by more than 2 points in the expected QoE score.

Main insight for viewport prediction error. Increased levels of VPE may result in reduced viewport quality and QoE. The VPE does not introduce further degradation when subjected to low-performance networks. The tiling scheme has the potential to highly affect QoE when considering intermediate levels of prediction error and network performance.



(a) VPE 0% VS bitrate (b) VPE 50% VS bitrate (c) VPE 100% VS bitrate

(d) VPE 0% VS QoE (e) VPE 50% VS QoE (f) VPE 100% VS QoE

Fig. 6. The effects of the viewport prediction error and tiling scheme on playout performance and QoE.

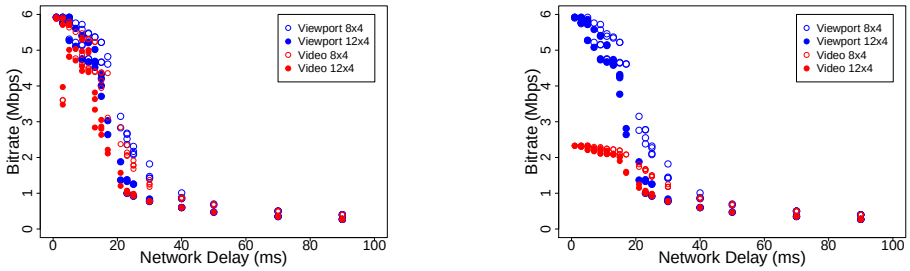
5.2 Per Tile Rate Adaptation Heuristics

As discussed in Section 2, the tile-based rate adaptation algorithm is crucial for achieving a suitable balance between playout performance and network bandwidth consumption. Although VR-EXP can be extended to encompass several strategies, in this section we focus on two distinct approaches, namely the Full Delivery (FD) [25] and the Full Delivery Basic (FDB) [12]. Recall that both approaches request the tiles inside the viewport in the highest possible quality representation. The main difference between them is that, depending on the available bandwidth, the FD method attempts to increase the bitrate for all the tiles, including the ones outside the viewport. Conversely, the FDB approach does not increase the quality representation for tiles outside the viewport, regardless of the available bandwidth.

Figures 7a and 7b show the relationship between the measured viewport bitrate (blue) and the entire video bitrate (red), when subjected to variable network performance conditions. The difference between FD and FDB is more noticeable when the delay is lower than 20 ms. In this case, FD benefits from the available network performance to maximize the quality representation of the entire video.

One key advantage of the FD approach is its natural protection against viewport prediction errors, at the cost of increased bandwidth consumption. On the other hand, when considering methods for viewport prediction with low error rates, the FDB method may represent a better choice as it will maintain good levels of QoE while avoiding bandwidth waste. For intermediate network delay (between 20 and 40 ms), both methods perform similarly, because the network performance is sufficient to accommodate only the viewport in high quality. Finally, for a network delay higher than 40 ms, there is no room for increasing the quality representation at all, and both strategies present equivalent performance.

Main insight for rate adaptation heuristics. The FD heuristic provides excellent protection against viewport prediction errors at the cost of increased bandwidth consumption. If combined with low-error viewport prediction algorithms, FDB may potentially lead to reduced bandwidth consumption.



(a) Full Delivery (b) Full Delivery Basic

Fig. 7. Dynamic rate adaptation heuristics: FD and FDB.

5.3 Multithreaded Tile Downloading

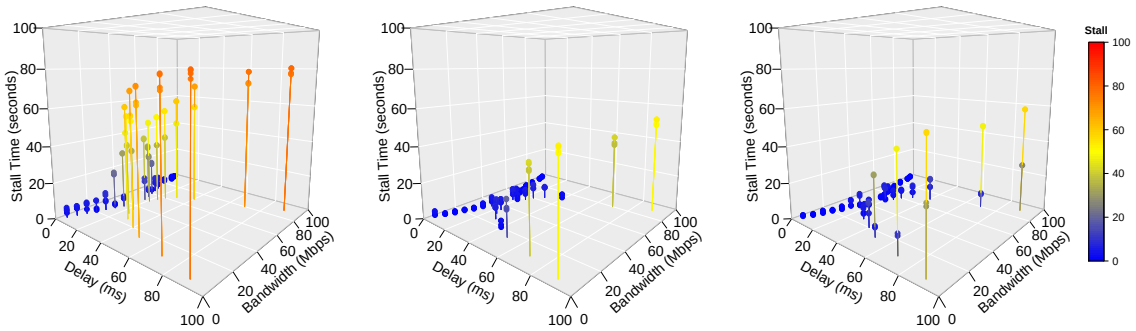
As discussed in Section 3, the network delay is the QoS metric that affects video playout performance the most. The reason is that high levels of network delay, when combined with both short video segments and tiling scheme overhead, limit the download throughput. Multithreaded tile request methods can improve the VR video playout performance by reducing the stall time. As shown in Figure 8b, when using six threads it is possible to dramatically reduce the VR video stall time. Basically, when compared to the single thread approach (Figure 8a), the use of six threads enables handling twice as much network delay (from 20 ms to 40 ms) while maintaining the same level of stall time. When resorting to ten threads for tile downloading (Figure 8c) it was possible to slightly reduce the stalling time, especially when considering VR videos using the 8x4 tiling scheme (as discussed next).

Figures 8e and 8f depict the effects of the multithreaded approach in the QoE score. When compared to the single thread (Figure 8d), the multithreaded approach is able to increase the QoE score in up to 1.5 points when the delay is higher than 20 ms. However, for network delays higher than 80 ms, the QoE is completely degraded, regardless of the available bandwidth and the use of multithreaded approaches.

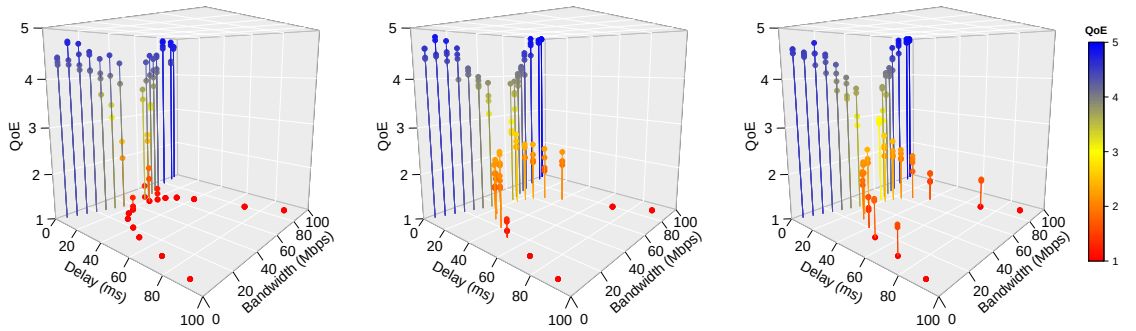
Figure 9 shows the effect of the multithreaded approach on distinct tiling schemes (*i.e.*, 8x4 and 12x4). When considering a network delay of 40 ms, the six threads approach outperforms the single thread by reducing the stall time from 60 to 5 seconds (approximately) (Figures 8a and 8b). The experiment with six threads resulted in similar results for both tiling schemes, with a slight advantage to the 8x4 one. In turn, the ten-thread experiment variation (Figure 8c) led to an

additional reduction of the stall time for the 8x4 scheme, but not for the 12x4, which presented roughly the same results when compared to the six-thread experiment.

Main insight for multithreaded tile downloading. Multithreaded tile fetching can dramatically reduce the stall time and increase the QoE score for intermediate levels of network performance. However, it does not provide noticeable improvements in QoE for either high or low network performance.

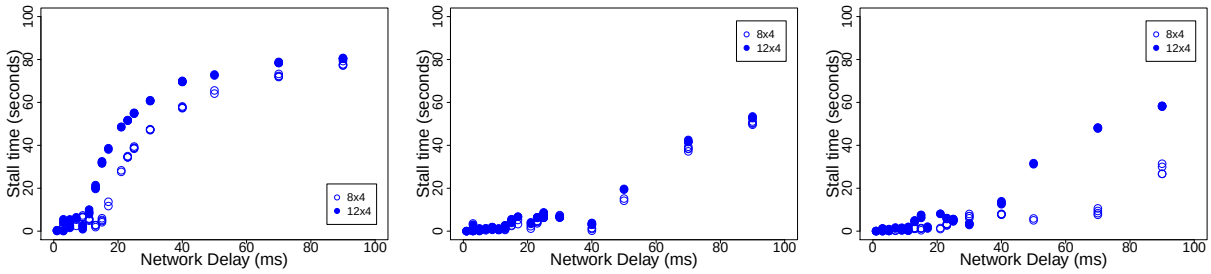


(a) Single thread - stall time (b) Six threads - stall time (c) Ten threads - stall time



(d) Single thread - QoE (e) Six threads - QoE (f) Ten threads - QoE

Fig. 8. Multi-thread effect on VR video stall time.



(a) Single thread - stall time (b) Six threads - stall time (c) Ten threads - stall time

Fig. 9. Multi-thread: stall time and tiling scheme VS network delay.

5.4 Buffer Size and Viewport Prediction Error

The evaluation carried out earlier in this section has focused on evaluating the effects of each VR video optimization technique on VR video playout performance and QoE. Aiming to further explore the interplay among different VR video optimization techniques, in this experiment, we evaluate a set of four optimization aspects simultaneously, namely variable viewport scheme, variable viewport prediction error, variable buffer size, and the FDB rate adaptation approach. Additionally, instead of evaluating how the optimization techniques perform when subjected to distinct network performance conditions, in this evaluation we vary the network conditions within the VR video session. Table 5 shows ten distinct combinations of network performance indicators that were randomly selected within the range for each QoS metric (as discussed in Section 4). A particular VR video session lasts for 60 seconds, where each network performance configuration lasts for 6 seconds, starting with the configuration ID 1 up to the ID 10. The main objective of this experiment is to evaluate the interplay between multiple VR video optimization approaches while subjected to highly variable network performance conditions. To provide a generalized analysis, the results presented in Figure 10 represent the averaged values when considering the entire VR video dataset. Therefore, the error bars, in this case, represent the min-max range for each histogram bin.

Table 5. Network performance indicators within a 60-second-long VR video session

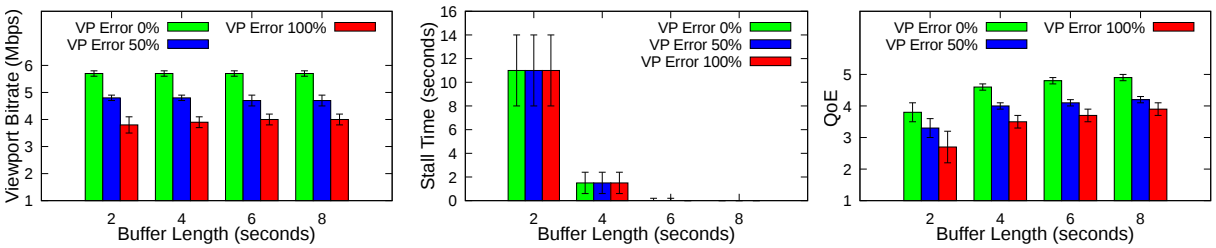
Conf. ID	Delay (ms)	Bandwidth (Mbps)
1	1	74
2	4	38
3	55	31
4	2	60
5	4	54
6	95	8
7	6	22
8	1	84
9	49	19
10	87	7

Figure 10a shows the average quality observed for the viewport when streaming VR videos subjected to variable buffer size and viewport prediction error rates. As discussed in Section 2, for most state-of-the-art viewport prediction algorithms, the bigger the buffer size, the higher the prediction error rate. Aiming at evaluating a broad range of scenarios, in the analysis presented in Figure 10a used a full factorial experiment design considering different values for buffer size and error rate. The obtained results indicate that the viewport prediction error greatly affects the viewport bitrate, while the buffer size itself does not have noticeable influence on it.

Figure 10b shows that the increased buffer size was able to dramatically reduce the stall time. For example, when considering a playout buffer dimensioned for 4 seconds of video, the stall time drops from 11 seconds to less than 2 seconds (on average). Furthermore, when increasing the buffer to 8 seconds, it was possible to completely eliminate the stall time. However, as discussed in Section 2, most state-of-the-art viewport prediction algorithms experiment sudden accuracy drop when increasing the playout buffer size. Hence, the effective analysis of the interplay between buffer size and viewport error must be done through the evaluation of the QoE indicator, since the QoE score will simultaneously consider both playout performance metrics. Figure 10c shows that, when using 8 seconds of playout buffer, the worst case scenario for the QoE score (*i.e.*, viewport prediction error of 100%) performs on par with the best case scenario of the 2 seconds buffer (*i.e.*, viewport prediction error of 0%). Furthermore, due to the human randomness, prediction algorithms may present low accuracy even when considering small buffers (*e.g.*, 2 sec). Therefore, using higher

values for dimensioning the playout buffer (*e.g.*, 8 sec) will probably outperform smaller buffers setups in most cases.

Main insight for mixed buffer size and prediction error. When dealing with realistic performance levels, increasing the playout buffer size may potentially lead to a better QoE score, even considering the likely increase in the prediction error.



(a) Viewport average bitrate (b) Stall time (c) QoE

Fig. 10. The influence of multiple VR video optimization techniques on VR video streaming playout performance and QoE.

6 CONCLUSION

VR video streaming applications are growing fast. To cope with the huge demand for network resources, both the scientific community and the industry have proposed optimization techniques for VR videos. However, the complex interplay between VR video optimization techniques and variable network conditions challenges developers of VR video solutions, as this interaction is neither trivial nor has it been properly investigated. Additionally, a publicly-available solution to provide a reproducible and in-depth evaluation of the VR video realm is still missing.

To address this problem, we proposed VR-EXP, an open-source platform for evaluating adaptive VR video streaming that encompasses various optimization techniques and allows for network performance conditions to be varied. To support realistic evaluation, we provide a 4G/LTE performance dataset comprised of multiple network performance metrics. Employing VR-EXP, along with realistic datasets, we have produced an extensive assessment that examines the performance of several state-of-the-art optimization techniques when subjected to variable network conditions. The results obtained evidence that the relationship between different optimization techniques for video VR optimization is not trivial. Mainly, because certain combinations can benefit one aspect of reproduction and impair others. For example, the increased buffer size, combined with the FDB approach, may lead to increased viewport prediction error. In this case, the viewport bitrate will be degraded and the stall time will be reduced. By combining an objective assessment of VR video streaming playout performance and a comprehensive QoE model, VR-EXP allowed pinpointing the components of the VR video ecosystem that most affect the performance of VR video playout and, ultimately, QoE.

The benefits of this work are twofold. From the VR video developers' perspective, we expect to contribute a useful approach to conducting a precise and realistic performance evaluation of novel optimization techniques. In turn, from the mobile operator's perspective, we expect VR-EXP to be a valuable tool for supporting investigations aimed at understanding and predicting how variable network conditions impact VR video performance and QoE delivered to their end-users.

7 ACKNOWLEDGMENTS

This research was performed partially within the project G025615N "Optimized source coding for multiple terminals in self-organizing networks" from the fund for Scientific Research-Flanders (FWO-V). Maria Torres Vega is funded by a grant of the Research Foundation - Flanders (FWO). This work was also partially funded by CAPES, CNPq, FAPERGS and IFSul.

REFERENCES

[1] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning Video ABR Algorithms to Network Conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. ACM, New York, NY, USA, 44–58. https://doi.org/10.1145/3230543.3230558

[2] Mathias Almquist, Viktor Almquist, Vengatanathan Krishnamoorthi, Niklas Carlsson, and Derek Eager. 2018. The Prefetch Aggressiveness Tradeoff in 360&Deg; Video Streaming. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 258–269. https://doi.org/10.1145/3204949.3204970

[3] Jill Boyce, Elena Alshina, Adeel Abbas, and Yan Ye. 2017. JVET common test conditions and evaluation procedures for 360 video. *Joint Video Exploration Team of ITU-T SG* 16 (2017).

[4] Zhenzhong Chen, Yiming Li, and Yingxue Zhang. 2018. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. *Signal Processing* 146 (2018), 66 – 78. https://doi.org/10.1016/j.sigpro.2018.01.004

[5] Cisco. 2018. *Cisco Visual Networking Index: Forecast and Trends, 2017âĂŞ2022*. Technical Report. Cisco Systems.

[6] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2017.7996611

[7] R. I. T. da Costa Filho, W. Lautenschlager, N. Kagami, V. Roesler, and L. P. Gaspary. 2016. Network Fortune Cookie: Using Network Measurements to Predict Video Streaming Performance and QoE. In *2016 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOCOM.2016.7842022

[8] Roberto Irajá Tavares da Costa Filho, Marcelo Caggiani Luizelli, Maria Torres Vega, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Filip De Turck, and Luciano Paschoal Gaspary. 2018. Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 270–283. https://doi.org/10.1145/3204949.3204966

[9] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 513–526. https://doi.org/10.1145/2987443.2987459

[10] Glederson Lessa dos Santos, Vinicius Tavares Guimaraes, Jorge Guedes Silveira, Alexandre T Vieira, Jose Augusto de Oliveira Neto, RIT da Costa, and Ricardo Balbinot. 2007. UAMA: a unified architecture for active measurements in IP networks; end-to-end objetive quality indicators. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 246–253.

[11] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360 Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM, New York, NY, USA, 67–72. https://doi.org/10.1145/3083165.3083180

[12] Mario Graf, Christian Timmerer, and Christopher Mueller. 2017. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 261–271. https://doi.org/10.1145/3083187.3084016

[13] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Favor: Fine-grained Video Rate Adaptation. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 64–75. https://doi.org/10.1145/3204949.3204957

[14] M. Hosseini and V. Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. 107–110. https://doi.org/10.1109/ISM.2016.0028

[15] Xueshi Hou, Sujit Dey, Jianzhong Zhang, and Madhukar Budagavi. 2018. Predictive View Generation to Enable Mobile 360-degree and VR Experiences. In *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network '18)*. ACM, New York, NY, USA, 20–26. https://doi.org/10.1145/3229625.3229629

[16] H. Hristova, X. Corbillon, G. Simon, V. Swaminathan, and A. Devlic. 2018. Heterogeneous Spatial Quality for Omnidirectional Video. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. 1–6. https://doi.org/10.1109/MMSP.2018.8547114

[17] E. Jeong, D. You, C. Hyun, B. Seo, N. Kim, D. H. Kim, and Y. H. Lee. 2018. Viewport Prediction Method of 360 VR Video Using Sound Localization Information. In *2018 Tenth International Conference on Ubiquitous and Future Networks*

(ICUFN). 679–681. https://doi.org/10.1109/ICUFN.2018.8436981

[18] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Stanford, CA, 1137–1145.

[19] L. Ma, Y. Xu, J. Sun, W. Huang, S. Xie, Y. Li, and N. Liu. 2018. Buffer Control in VR Video Transmission Over MMT System. In *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 1–5. https://doi.org/10.1109/BMSB.2018.8436817

[20] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[21] A. Morton. 2016. *RFC 7799 - Active and Passive Metrics and Methods (with Hybrid Types In-Between)*. Technical Report. IETF.

[22] T. C. Nguyen and J. Yun. 2018. Predictive Tile Selection for 360-Degree VR Video Streaming in Bandwidth-Limited Networks. *IEEE Communications Letters* 22, 9 (Sep. 2018), 1858–1861. https://doi.org/10.1109/LCOMM.2018.2848915

[23] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages. https://doi.org/10.1145/2818361

[24] S. Petrangeli, G. Simon, and V. Swaminathan. 2018. Trajectory-Based Viewport Prediction for 360-Degree Virtual Reality Videos. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 157–160. https://doi.org/10.1109/AIVR.2018.00033

[25] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*. ACM, New York, NY, USA, 306–314. https://doi.org/10.1145/3123266.3123453

[26] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 Video Delivery over Cellular Networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC '16)*. ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/2980055.2980056

[27] Iraj Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia* 18, 4 (2011), 62–67.

[28] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 123–137. https://doi.org/10.1145/3204949.3204953

[29] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524428

[30] K. Stangherlin, R. C. Filho, W. LautenschlÃďger, V. Guadagnin, L. Balbinot, R. Balbinot, and V. Roesler. 2011. One-way delay measurement in wired and wireless mobile full-mesh networks. In *2011 IEEE Wireless Communications and Networking Conference*. 1044–1049. https://doi.org/10.1109/WCNC.2011.5779279

[31] TELCO. 2018. Mobile Operators Market Share in Brazil. http://www.teleco.com.br/en/en_mshare.asp Accessed 14-January-2018.

[32] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.

[33] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, and T. D. HÃďmÃďlÃďinen. 2015. Kvazaar HEVC encoder for efficient intra coding. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1662–1665. https://doi.org/10.1109/ISCAS.2015.7168970

[34] VR-EXP. 2019. VR Video Streaming Experimentation Platform. https://github.com/rtcostaf/TOMM2019_VR-EXP Accessed 5-January-2019.

[35] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 193–198. https://doi.org/10.1145/3083187.3083210

[36] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 325–338. https://doi.org/10.1145/2785956.2787486

[37] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 325–338. https://doi.org/10.1145/2829988.2787486

[38] Chao Zhou, Zhenhua Li, Joe Osgood, and Yao Liu. 2018. On the Effectiveness of Offset Projections for 360-Degree Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 3s, Article 62 (June 2018), 24 pages. https://doi.org/10.1145/3209660