TECHNISCHE UNIVERSITÄT BERLIN

Modeling and Design of Novel QoE Management Strategies for Adaptive Video Streaming

vorgelegt von
Susanna Maria Schwarzmann, M.Sc.

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr. Manfred Hauswirth
Gutachter:	Prof. Dr. Thomas Zinner
Gutachter:	Prof. Dr. Stefan Schmid
Gutachter:	Prof. Dr. Oliver Hohlfeld

Tag der wissenschaftlichen Aussprache: 25. April 2022

Berlin 2022

Abstract

Today's Internet serves a huge variety of different applications with diverse and ever increasing demands on the underlying network. Among others, current trends towards immersive entertainment, like 8K video streaming or VR gaming, pose new challenges on end-to-end bandwidth volume and have stringent delay requirements. At the same time, the number of users is growing, as well as the users' expectations on the delivered service quality. As a consequence, delivering good Quality of Experience (QoE) becomes an ever more challenging task and – due to the steadily increasing number of providers – the satisfaction of subscribers and users is a substantial factor to remain competitive on the market. As a consequence, QoE management has emerged as a key research topic over the past years and constantly gains importance for several stakeholders in the Internet ecosystem. This monograph examines relevant research questions related to QoE management on the example of HTTP Adaptive Streaming (HAS), which is to date the application contributing the most to the global IP traffic.

One of the major challenges of QoE management is to understand the complex interplay of application- and network-specific parameters and their impact on the user satisfaction. The first part of this thesis shows how QoE-relevant performance metrics for HAS can efficiently be retrieved for a given – and potentially huge – input parameter space by means of analytical modeling. More specifically, we use an existing approach relying on discrete-time analysis, which models an HAS client's video buffer, and we extend it so to reflect the HAS-typical quality adaptation behavior. For given input network characteristics, such as the available bandwidth and its variation, as well as for various video- and player-specific settings, like the quality switching thresholds, the model yields probabilistic outputs for the video buffer's filling state. From that, all relevant HAS metrics, e.g., the stalling behavior and the delivered video quality, can be derived, allowing to efficiently tune HAS parameters in accordance with each other, so as to optimize the QoE.

The second part quantifies possible positive effects of using variable segment durations for HAS. Instead of relying on a content-agnostic video segmentation strategy with fixed segment durations, the variable approach – proposed by Netflix as shot-based encoding – takes the video content into account by segmenting the video at scene-cuts. This results in segments of different lengths, but promises to reduce the number of costly I-frames during the encoding and hence, to increase the encoding efficiency. However, no comparative study highlighting the impact of this technique on the HAS ecosystem has been conducted, yet. Thus, we first provide a broad investigation on the bitrate reduction that can be achieved with the variable approach. In a second step, we evaluate by means of a measurement study its impact on the streaming performance, taking into account three different adaptation heuristics. Our results show that variable segment durations can significantly reduce the bitrate requirements and – as a result – are capable of increasing the HAS QoE.

The third part of this thesis focuses on how mobile network operators (MNOs) can exploit new features provided by the 5G networking architecture, so to overcome current QoE monitoring limitations. More specifically, we propose to make use of Network Functions (NFs) which are introduced with 5G and dedicated for improved analytics, complex computations, and for the interaction with third parties, such as content providers. These capabilities enable a variety of potentials, like, for example, estimating the QoE by applying Machine Learning (ML) techniques. From the perspective of an MNO, we elaborate on the involved challenges of introducing such an ML-based QoE estimation in 5G networks and by means of a simulation-based feasibility study, we demonstrate that the QoE can reliably be estimated solely based on network KPIs. In this scope, we perform a quantitative comparison, addressing the estimation accuracy of different state-of-the-art regression techniques, and discuss them with respect to different relevant qualitative aspects.

Zusammenfassung

Das heutige Internet wird für eine Vielzahl verschiedenartiger Anwendungen genutzt. Diese haben zum einen sehr diverse Anforderungen an das zugrundeliegende Netz, welche zum anderen auch stetig steigen. Besonders stark ist dieser Trend bei den Unterhaltungsmedien zu beobachten. Ultrahochauflösendes Video Streaming, zum Beispiel in 8K, fordert immer höhere Bandbreiten ein, während Onlinespiele zusätzlich immer striktere Bedingungen bezüglich der maximal auftretenden Paketverzögerungen haben. Können diese Anforderungen vom Netz nicht erfüllt werden, so wird die vom Nutzer subjektiv wahrgenommene Qualität der Anwendung beeinträchtigt. Besipielsweise führt eine zu geringe Bandbreite im Fall von Video Streaming zu Wiedergabeunterbrechungen und schlechter Bildqualität. Wenn bei Voice-over-IP Telefonaten Paketverlust oder -verzögerung auftreten, so wird die Qualität der Sprachübertagung wesentlich gestört. Um die subjektive Wahrnehmung zu quantifizieren, hat sich in den letzten Jahren in der Forschung das Konzept „Quality of Experience“ (QoE) etabliert, welches den Grad der (Un)-zufriedenheit des Nutzers eines Services beschreibt. Dieser resultiert aus der Erfüllung der individuellen Erwartung an den Service.

Während zum einen die Erwartungshaltung der einzelnen Nutzer stetig dahingehend zunimmt, dass die Anwendungen stabil und ohne Qualitätseinbußen laufen, werden auch die Angebote selbst, wie zum Beispiel Video-on-Demand Plattformen, immer beliebter. So nimmt die tägliche Nutzung von Video Streaming, virtuellen Konferenzen oder auch Cloud Gaming weltweit zu. Diese Trends, gepaart mit den kontinuierlich steigenden Anforderungen heutiger Services an das Netz, stellen sowohl Anwendungs- als auch Netzanbieter vor große Herausforderungen. Zum einen muss die Infrastruktur entsprechend skaliert sein, so dass Applikationen flüssig laufen und den Nutzern eine größtmögliche Zufriedenheit geboten wird, um sie nicht an die Konkurrenz zu verlieren. Zum anderen aber müssen die Anbieter kostengünstig agieren, um wirtschaftlich rentabel zu bleiben, um so am stetig wachsendem Markt konkurrenzfähig zu sein.

Um diese Ziele zu vereinen, benötigt es intelligente Mechanismen auf Netz- und Anwendungsebene, die es erlauben, vorhandene Ressourcen möglichst effizient zu nutzen, aber dennoch den verschiedenen Anwendungsanforderungen gerecht zu werden. „Quality of Service“ (QoS) Management erlaubt es, relevante Netzwerkparameter wie zum Beispiel Durchsatz oder Paketverzögerungen zu messen und bei Bedarf diese Parameter mit Hilfe verschiedener Steuerungsmechanismen, wie zum Beispiel Bandbreitenreservierung für einzelne Verbindungen, zu verbessern. Zwar ist durch die Verbesserung der Netzwerkparameter von einer Verbesserung der Anwendungsqualität auszugehen, die genaue Auswirkung auf die vom Nutzer wahrgenommene Dienstgüte kann jedoch nur schwer quantifiziert werden.

Aus diesem Grund geht QoE Management – im Vergleich zu QoS Management – einen Schritt weiter und berücksichtigt bei der Umsetzung von Steuerungsmechanismen weitere Kontextinformationen, wie die spezifischen Anforderungen der laufenden Anwendung,

oder das vom Nutzer verwendete Endgerät und dessen Eigenschaften. Somit ermöglicht QoE Management eine gezieltere Anpassung des Netzwerks und der Anwendung, so dass vorhandene Ressourcen effizient genutzt werden, und die Nutzerzufriedenheit maximiert wird. Allerdings ist das eine sehr komplexe Aufgabe, gekoppelt mit einer Vielzahl an Herausforderungen, weshalb sich QoE Management in den letzten Jahren zu einem wichtigen Forschungsfeld entwickelt hat. Elementares Ziel ist es, leistungsfähige und optimierte Lösungen in Bezug auf die drei folgenden Grundbausteine zu entwickeln: Der erste ist die Modellierung und die Abschätzung von QoE. Dieser Baustein befasst sich damit, zu quantifizieren, wie Netzwerkparameter, anwendungsspezifische Einstellungen und die Erwartungshaltung des Nutzers dessen wahrgenommene Dienstgüte beeinflussen. Der zweite nötige Baustein, das QoE Monitoring, beschreibt die Fähigkeit, relevante Parameter im Netz und in der Anwendung, oder direkt die QoE, messen zu können. Schließlich, und das formt den dritten Grundbaustein, benötigt man noch Mechanismen, um das Netz und die Anwendung gezielt steuern zu können, um am Ende die QoE zu verbessern oder ein bestimmtes Level an Nutzerzufriedenheit garantieren zu können. Diese Arbeit adressiert die verschiedenen Herausforderungen von QoE Management, stellt unterschiedliche Lösungsansätze in Bezug auf alle drei Bausteine vor und evaluiert diese im Hinblick auf ihre Umsetzbarkeit und Leistungsfähigkeit.

Zunächst stellen wir ein analytisches Modell vor, welches auf Warteschlangentheorie basiert und in der Lage ist, QoE-relevante Metriken für adaptives Video Streaming effizient zu berechnen. Das Modell nimmt als Eingabe verschiedene netz-, video- und anwendungsspezifische Parameter entgegen. Dazu gehören unter anderem die vorhandene Bandbreite und deren Schwankung, die verfügbaren Videobitraten und die maximale Kapazität des Videopuffers. Die hohe Effizienz des Modells erlaubt es, mit überschaubarem Zeitaufwand eine Vielzahl an verschiedenen Parameterkombinationen zu testen und somit deren Zusammenspiel und Einfluss auf die QoE zu evaluieren. Es kann also benutzt werden, um die einzelnen Einstellungsparameter von adaptivem Streaming so zu optimieren, dass unter verschiedenen Netzwerkbedingungen die vom Nutzer erfahrene Dienstgüte maximiert wird.

Des Weiteren untersuchen wir in dieser Arbeit, inwiefern eine effizientere Strategie zur Videosegmentierung die QoE von Streaminganwendungen verbessern kann. Typischerweise wird ein Video für adaptives Streaming – ohne jegliche Berücksichtigung des Videoinhalts – in gleichlange Segmente von wenigen Sekunden Dauer unterteilt und jedes dieser Segmente mit verschiedenen Bitraten codiert. Während der Wiedergabe kann somit die Qualität dynamisch an die aktuell verfügbare Bandbreite angepasst werden. Allerdings verringert eine solche, von der Videostruktur unabhängige Segmentierung, die Effizienz beim Encoding. Berücksichtigt man jedoch die Charakteristiken des zugrundeliegenden Videos und unterteilt das Video an den Stellen, an welchen ein Szenenwechsel stattfindet, dann kann das die Effizienz des Encodings steigern, liefert aber Segmente von variabler Dauer. Zunächst quantifizieren wir unter Einbeziehung verschiedener encodingspezifischer Einflussfaktoren, wie viel Bitrate der variable Ansatz im Vergleich zum konventionellen Ansatz einsparen kann. Wir zeigen, dass das Potenzial von variablen Segmentdauern am stärksten vom gewählten Videoclip selbst abhängt, und dass die Videobitrate bei gleichbleibender Qualität signifikant

reduziert werden kann. In einem zweiten Schritt evaluieren wir die Leistungsfähigkeit von variablen Segmentdauern beim Video Streaming selbst. Wir stellen fest, dass die reduzierte Bitrate vor allem in solchen Situationen die QoE deutlich verbessern kann, in denen nur wenig Bandbreite zur Verfügung steht.

Der dritte Teil dieser Arbeit bezieht sich auf den Baustein QoE Monitoring und untersucht, wie sich Netzwerkbetreiber maschinelles Lernen zu Nutze machen können, um die QoE basierend auf Telemetriedaten aus dem Netzwerk zu schätzen. Die 5G Netzwerkarchitektur führt neue, dedizierte Netzfunktionen für die Datenanalyse, komplexe Berechnungen, sowie für den Informationsaustausch mit Dritten, zum Beispiel einem Inhaltsanbieter, ein. Diese Netzfunktionen eröffnen eine Vielzahl neuartiger Möglichkeiten, wie zum Beispiel eine zuverlässige Schätzung der QoE im Netz durch die Integration von maschinellem Lernen. Zunächst diskutieren wir die Herausforderungen eines solchen Ansatzes in 5G Netzen und untersuchen die Anwendbarkeit und Leistungsfähigkeit verschiedener Regressionstechniken. Unsere Ergebnisse zeigen, dass gängige Techniken des maschinellen Lernens für QoE Monitoring, welches rein auf Telemetriedaten aus dem Netz beruht, mit hoher Effizienz und Genauigkeit eingesetzt werden können. Der vorgestellte Ansatz kann somit die Grundlage für ein wirkungsvolles QoE Management in zukünftigen Netzwerken bilden.

Danksagung

Mit der Fertigstellung meiner Dissertation blicke ich nun auf einige sehr prägende Jahre zurück. Eine Zeit, in der mich viele Menschen begleitet und unterstützt haben, wofür ich mich von Herzen explizit bei allen – auch nicht namentlich genannten – bedanken möchte.

Mein größter Dank gebührt meinem Doktorvater Prof. Dr. Thomas Zinner für die einzigartig vertrauensvolle Zusammenarbeit. Er hat mich 2017 in Würzburg am LS3 in seine Forschungsgruppe aufgenommen und damit den Grundstein für meine wissenschaftliche Laufbahn gelegt. Seitdem hat er mir mit unentwegter Geduld die Techniken gescheiten wissenschaftlichen Arbeitens und Schreibens mit auf den Weg gegeben und mir Know-how und Skills vermittelt wie kein Zweiter. Ich könnte hundert Dinge aufzählen, für die ich mich gerne bei ihm bedanken möchte, vor allem aber möchte ich seine enorme persönliche Unterstützung nennen. Nur durch seine Ermutigung habe ich einige Chancen ergriffen, über die ich heute überaus dankbar bin.

Darüber hinaus möchte ich mich bei Prof. Dr. Manfred Hauswirth, der den Prüfungsvorsitz übernommen hat, bedanken, sowie bei Prof. Dr. Oliver Hohlfeld für die Begutachtung meiner Arbeit. An dieser Stelle möchte ich auch meinen weiteren Gutachter Prof. Dr. Stefan Schmid nennen, mit dem ich noch einige Monate in Berlin zusammenarbeiten durfte, nachdem er die Leitung der INET Fachgruppe übernommen hat. Ihm möchte ich für die schöne – wenn auch leider nur kurze – gemeinsame Zeit und die stets vertrauensvolle Zusammenarbeit danken.

Teile dieser Arbeit sind im Rahmen des vom BMBF geförderten Software Campus entstanden. Dadurch ergab sich eine sehr herzliche Kooperation mit Dr. Clarissa Marquezan und Dr. Riccardo Trivisonno vom Huawei Munich Research Center. Ich bedanke mich bei den beiden dafür, dass sie stets wertvolle Ansprechpartner für mich waren. Durch ihren tatkräftigen Support und ihren hilfreichen Input haben sie einen signifikanten Beitrag zum Erfolg meiner Doktorarbeit geleistet.

Mein herzlicher Dank geht auch an Prof. Dr. Phuoc Tran-Gia, der mir damals den Beginn der Promotion in Würzburg ermöglicht hat, sowie an Prof. Dr. Tobias Hoßfeld und die gesamte Gruppe vom LS3. Durch die freundschaftliche Atmosphäre am Lehrstuhl und die vielen Aktivitäten außerhalb der Arbeit, blicke ich auf eine sehr schöne Zeit in Würzburg zurück. Insbesondere möchte ich mich bei Alexej Grigorjew bedanken, der nicht nur der Arbeitskollege war, der immer eine Lösung für meine technischen Problemchen hatte, sondern auch mein engster Freund in Würzburg seit Studienbeginn ist. Vielen Dank auch an Kathrin Borchert, die als Bürokollegin immer für die nötige Heiterkeit gesorgt hat, und an Frank Loh, der immer für eine entspannte Kaffeepause auf der Dachterrasse zu haben war.

Nach eineinhalb Jahren Promotion in Würzburg wechselte ich an die TU Berlin und wurde dort Teil der Fachgruppe INET. Dadurch durfte ich viele inspirierende und weltoffene Menschen kennen lernen. Das gilt sowohl für die „alte“ INET Gruppe unter ehemaliger

Leitung von Prof. Dr. Anja Feldmann, als auch für die „neue“ INET Gruppe unter Leitung von Prof. Dr. Stefan Schmid. Ich bedanke mich herzlich bei allen Berliner Kolleginnen und Kollegen für das aufgeschlossene Miteinander und die unvergessliche Zeit. Ganz besonders möchte ich hier Niklas Semmler und Matthias Rost nennen. Durch euch haben selbst die ganz langen Abende im Büro – kurz vor wichtigen Deadlines – immer viel Spaß gemacht. Danke für die hilfreichen Diskussionen, eure Unterstützung, die gemeinsamen Reisen und alle sportlichen, abenteuerlichen und entspannten Aktivitäten außerhalb der Arbeit.

Keine der Publikationen ist im Alleingang entstanden und alle Ko-Autoren haben einen wesentlichen Teil zum erfolgreichen Abschluss dieser Arbeit beigetragen. Vielen Dank für die stets gute Zusammenarbeit und die vielfältige Unterstützung. An dieser Stelle möchte ich auch allen studentischen Hilfskräften danken, die mich mit viel Engagement unterstützt haben: Huiran Liu, Hendrik von Kiedrowski, Leonie Reichert, Lukas Beierlieb, Marcin Bosk und Paula Breitbach. Ebenso danke ich allen Studenten, die ich Rahmen von Projekt- und Abschlussarbeiten betreuen durfte. Ganz besonders Nick Hainke für seine außerordentliche Geduld und Initiative beim Videocodieren. Mein Dank geht auch an alle Admins, die mir bei jeglichen Setups viel unter die Arme gegriffen haben, allen voran Sarah Dierenfeld. Ferner geht mein Dank an Alison Wichmann und Birgit Hohmeier-Touré, die bei allen nicht-technischen Belangen stets eine ausgesprochen große Hilfe waren.

Vor allem während der letzten Phasen der Promotion habe ich reichlich Unterstützung von vielen Seiten bekommen. Das bezieht sich insbesondere auf Feedback zum Vortrag und dem Korrekturlesen der Arbeit. Vielen Dank an die gesamte „Self-adjusting Networks Reading Group“, sowie an Alexej Grigorjew, Laura Schwarzmann, Marc Herrmann, Marija Gajic, Matthias Rost, Michael Jarschel, Niklas Semmler und Stanislav Lange.

Schließlich möchte ich noch Dankesworte an meine Familie und alle Freunde, die immer für den nötigen Ausgleich gesorgt haben, richten. Allen voran danke ich meinen Eltern Maria und Toni, die mich stets auf meinem Weg begleitet haben und mir bei jeder Herausforderung zur Seite standen. Danke auch an meine Geschwister Katharina, Michael, Laura und Felix, die immer für mich da sind und auf die ich mich jederzeit hundertprozentig verlassen kann.

Publications

In the following, a list of all works co-authored during my time as a PhD student is given. All my collaborators are among my co-authors.

International Conferences and Workshops

Evaluation of the Benefits of Variable Segment Durations for Adaptive Streaming
Susanna Schwarzmann, Thomas Zinner, Stefan Geissler, Christian Sieber.
10th International Conference on Quality of Multimedia Experience (QoMEX), 2018.

Computing Qoe-relevant Adaptive Video Streaming Metrics Using Discrete-time Analysis.
Susanna Schwarzmann, Paula Breitbach, Thomas Zinner.
22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2019.

Modeling Adaptive Video Streaming Using Discrete-time Analysis.
Susanna Schwarzmann, Paula Breitbach, Thomas Zinner, Matthias Rost.
31st International Teletraffic Congress (ITC 31), 2019.

Estimating Video Streaming QoE in the 5G Architecture Using Machine Learning.
Susanna Schwarzmann, Clarissa Marquezan, Marcin Bosk, Huiran Liu, Riccardo Trivisonno, Thomas Zinner.
Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks, 2019.

Comparing Fixed and Variable Segment Durations for Adaptive Video Streaming: A Holistic Analysis.
Susanna Schwarzmann, Nick Hainke, Thomas Zinner, Christian Sieber, Werner Robitza, Alexander Raake.
Proceedings of the 11th ACM Multimedia Systems Conference, 2020.

Accuracy vs. Cost Trade-off for Machine Learning Based QoE Estimation in 5G Networks.
Susanna Schwarzmann, Clarissa Marquezan, Riccardo Trivisonno, Shinichi Nakajima, Thomas Zinner.
IEEE International Conference on Communications (ICC), 2020.

Linking QoE and Performance Models for DASH-based Video Streaming.
Susanna Schwarzmann, Thomas Zinner.
6th IEEE Conference on Network Softwarization (NetSoft), 2020.

HTBQueue: A Hierarchical Token Bucket Implementation for the OMNeT++INET Framework.
Marcin Bosk, Marija Gajić, Susanna Schwarzmann, Stanislav Lange, Thomas Zinner.
Proceedings of the 8th OMNeT++ Community Summit, 2021.

Using 5G QoS Mechanisms to Achieve QoE-Aware Resource Allocation.
Marcin Bosk, Marija Gajić, Susanna Schwarzmann, Stanislav Lange, Riccardo Trivisonno, Clarissa Marquezan, Thomas Zinner
17th International Conference on Network and Service Management (CNSM), 2021.

International Journals

Quantitative Comparison of Application–network Interaction: A Case Study of Adaptive Video Streaming.
Susanna Schwarzmann, Thomas Zinner, Ognjen Dobrijevic.
Quality and User Experience, 2017.

Scalable Application-and User-aware Resource Allocation in Enterprise Networks Using End-host Pacing.
Christian Sieber, Susanna Schwarzmann, Andreas Blenk, Thomas Zinner, Wolfgang Kellerer.
ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), 2020.

ML-based QoE Estimation in 5G Networks Using Different Regression Techniques
Susanna Schwarzmann, Clarissa Marquezan, Riccardo Trivisonno, Shinichi Nakajima, Vincent Barriac, Thomas Zinner.
IEEE Transactions on Network and Service Management (TNSM), 2022.

Book Chapters

QoE Management for Future Networks.
Raimund Schatz, Susanna Schwarzmann, Thomas Zinner, Ognjen Dobrijevic, Eirini Liotou, Peter Pocta, Sabina Barakovic, Jasmina Barakovic Husic, Lea Skorin-Kapov.
Autonomous Control for a Reliable Internet of Services, 2018.

Big-data Helps SDN to Improve Application Specific Quality of Service.
Susanna Schwarzmann, Andreas Blenk, Ognjen Dobrijevic, Michael Jarschel, Andreas Hotho, Thomas Zinner, Florian Wamser, 2018.

AI in 5G Networks: Challenges and Use Cases.
Stanislav Lange, Susanna Schwarzmann, Marija Gajić, Thomas Zinner, Frank A Kraemer.
Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning, 2021.

Contents

1 Introduction

Today's Internet serves an ever growing number and variety of applications. While their requirements on the network are highly diverse, they are at the same time constantly increasing. This trend is particularly prominent in the multimedia and entertainment domain. For example, while ultra-high definition (UHD) video streaming demands more and more bandwidth capacity, online cloud gaming applications additionally have stringent demands in terms of packet delays. If the requirements cannot be fully met, the application quality – as perceived by the user – is degraded. For example, insufficient bandwidth causes interruptions during a video stream and results in poor visual quality. Increased packet delays lead to lacks in online games and to delayed audio playback at the receiver side of a VoIP call. In order to quantify the impact of network impairments on the application quality, as subjectively perceived by the end-user, the concept of *Quality of Experience (QoE)* has emerged as an important research topic during the last decade. It reflects the degree of delight or annoyance when using a specific service and is highly depending on the personal expectation a user has on that service.

While the expectations of the individual users on good service quality are steadily increasing, the market of Internet service offers is expanding simultaneously. For example, due the raising popularity of Video On Demand (VoD) platforms, conventional media technologies, such as broadcast TV, are step by step replaced. The modern working environment currently experiences a shift towards more flexibility, thus intensifying the usage of remote-office, cloud computing, and online conferencing solutions. These trends, combined with the steadily increasing network requirements of novel applications, pose huge challenges to

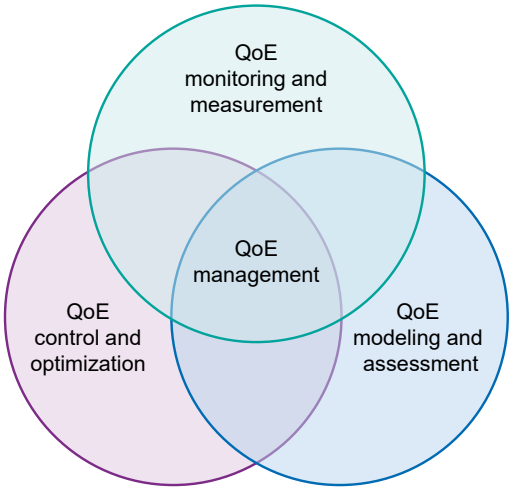


Figure 1.1: Illustration of the three key building blocks for QoE management [19].

both, application providers (APs) and network providers (NPs). Providing a good QoE to the user is an important business incentive for these stakeholders in the Internet ecosystem to reduce the ever present risk of customer churn [16]. While this can be achieved by deploying high-performance infrastructures and by scaling services so to achieve high quality experiences, NPs and APs also have to operate their networks and services economically in order to stay competitive on the expanding market [17]. Therefore, the crux is to exploit network resources and application-level mechanisms as efficiently as possible to achieve both, customer loyalty by delivering a good QoE plus being economically viable by profitable service operation.

In order to meet these conflicting objectives, it needs intelligent mechanisms on application- and network-layer, which allow to use the available resources as efficiently as possible, whilst still fulfilling the applications' requirements. The concept *Quality of Service* (QoS) refers to the monitoring of network parameters, such as throughput or latency, which can trigger the execution of targeted network control actions. That is, for example, adjusting a flow's guaranteed bitrate or prioritizing certain traffic. Although an improvement of the network parameters is associated with an improvement of the service quality, the specific implications on the quality – as subjectively perceived by the user – cannot be quantified exactly, due to the complex and service-specific relationship between QoS and QoE.

QoE management goes one step beyond and considers additional context factors such as the used applications, their specific requirements, or the user's end-device. While QoE management is promising in terms of meeting the trade-off between economic aspects and application performance in a service- and user-centric manner, it is challenging and involves complex tasks. Although there is no standardized definition of them, there is a consensus in the QoE research community that the QoE management process can be broken down to three major building blocks [18, 19, 20], as illustrated in Figure 1.1.

1. **QoE modeling and assessment**: It is essential to understand the requirements of given applications and how disturbances affect the QoE. The QoE modeling and assessment component describes the creation of predictive mathematical models, which allow to map how measurable parameters, i.e., network- or application-related key performance indicators (KPIs), affect the user perceived quality. These generic relationships between measurable parameters and QoE are a fundamental step towards understanding QoE [18] and are crucial to derive appropriate control actions.

2. **QoE monitoring and measurement**: Monitoring refers to the collection of data and information. This can be network- or application-related KPIs, such as throughput, packet loss, terminal capabilities, or video bitrate, which are used along with QoE models to retrieve the QoE. Other than that, this building block can also involve the collection of explicit user ratings, e.g., via surveys, which would give the best view of the user satisfaction.

3. **QoE control and optimization**: This component describes the adaptation and optimization via dynamic control actions on network- and application-level, potentially triggered by QoE monitoring activities. Those control actions can be short-termed, e.g., updated per-flow bandwidth guarantees or traffic prioritization, or long-termed, such as switching to an improved voice codec for VoIP telephony.

Each one of these building blocks is essential for QoE management, as no proper control actions could be performed without knowing the current state of the application performance and the possible implications of implemented control actions. Furthermore, each of the building blocks brings its own challenges and their specific realizations are still topic of research, among others because they constantly need to adapt to the dynamics of the Internet ecosystem. New applications like VR gaming, but also adaptations to existing services, such as the deployment of novel voice or video codecs, require new mapping functions between QoS and QoE used for *modeling QoE*. With the ever increasing share of end-to-end encrypted traffic, inspecting packet payloads, e.g., to retrieve information such as audio or video bitrate from network flows, is no longer applicable and NPs need to find *QoE monitoring* alternatives [21]. Advanced resource control mechanisms introduced with next generation networks, like Network Slicing or QoS-flows in 5G, offer new possibilities for *QoE control* in the network, but due to the recent 5G deployment, field studies and guidelines on how to optimally exploit them for QoE-aware networking are still missing.

One major flaw which hinders to fully exploit the potential of QoE management – and thus a highly relevant challenge – is the missing signaling between the network and the applications, allowing to exchange relevant information [18]. The positive effects of application-network collaboration have been shown in several studies and proposals [22, 23, 24, 25, 26]. But still, the AP and the NP are stakeholders, that typically can only operate independently from each other within the Internet ecosystem. This is due to their different vantage points and bounded control capabilities. For instance, the AP can monitor application performance metrics and adapt application-specific settings, but without solid awareness of the underlying network characteristics. Similarly, the NP has capabilities to monitor and control the

network, but it is not fully aware of the performance of running applications or their specific settings. Although the network KPIs collected via QoS monitoring allow an estimate of the QoE, more reliable information of the application performance would increase the effectiveness of QoE-aware adjustments. Collaborations between APs and NPs can help to overcome such issues, but are often hindered by practical obstacles, such as privacy concerns or network neutrality [26], as well as missing interfaces and network entities [18].

The limited system view and bounded control capabilities of APs and NPs increase the complexity of QoE management for either side, as the QoE delivered to the user is dictated by the specifics and requirements of the application, as well as by the underlying network and the degree to which it can satisfy the application's needs. However, the combination of all of these possible settings and conditions form such a huge problem space, that even if any conceivable information about the application and the network conditions would be available, it is still challenging to retrieve control actions that result in an optimized system performance. This is due to the missing holistic understanding of services. While controlled measurements and simulation activities allow to study certain use-cases, scalable approaches for generating such an overall understanding are still missing.

1.1 Research Questions

This work proposes novel strategies to overcome some of the major limitations of QoE management. On the example of HTTP Adaptive Streaming (HAS), which is one of the most widely used applications to date and which constitutes the majority of the overall global IP traffic volume [27], we examine how to better exploit the potential of QoE management. Thereby, we take into account the perspective of both stakeholders within the HAS ecosystem, i.e., the NP and the AP, and we cover all of the three key building blocks while elaborating on the following research questions:

RQ1 **How to efficiently build up a holistic understanding of the complex interplay between application- and network-specific parameters and their impact on QoE?** The AP has a huge range of capabilities to tune its HAS system so as to achieve a high level of satisfaction for the end user. That is, for example, the number of provided quality levels and their bitrate characteristics, threshold settings to define which quality level to select, or buffer-dependent presets for starting and pausing the playback. However, these settings need to be properly adjusted in accordance to each other on the one hand, and tuned to the underlying network conditions on the other hand. For example, higher bandwidth variations, as observed in mobile environments, require different settings of the control parameters compared to wired networks, where the bandwidth capacity is in general higher and more stable. The question on how to understand all relevant inter-dependencies relates to the QoE modeling and assessment building block and it is especially relevant for the AP to adjust its settings as good as possible, but also relevant for the NP to derive the requirements imposed on its network.

RQ2 **Does a more efficient video segmentation technique allow for further optimizing the HAS ecosystem, so as to increase the system efficiency and QoE?**
By splitting videos into short segments of equal duration and encoding each of them with a set of different bitrates, HAS allows to dynamically adapt the video quality during playback. However, the segmentation process causes additional encoding overhead, that is, in order to achieve the same visual quality for an HAS-prepared video, a higher bitrate is required compared to the unsegmented version of that video. By allowing segments of variable duration, this additional overhead can be reduced. In how far this condition can be exploited to achieve a generic improvement to the HAS ecosystem, is a question related to the QoE optimization building block. While the AP can potentially profit from this approach by providing a better visual quality with the same bitrate requirements, the NP's traffic volume could be decreased, although delivering the same quality.

RQ3 **How can new features of the 5th generation of mobile networks help to overcome the QoE monitoring limitations of today's networks?**
The missing interaction between AP and NP is a key limitation to efficient QoE management and is partially caused by practical issues, such as missing interfaces and network entities. This obstacle is eliminated in the 5G networking architecture, which provides a new network function (NF), dedicated for the information exchange between the network operator and third parties, such as an AP. Combined with another newly introduced NF, which leverages advanced data analytics in the network, 5G offers innovative opportunities for QoE monitoring. While this can potentially be a game changer towards user-centric network management, the question remains how to systematically exploit the provided features and how to define their specifics, which is not part of ongoing 5G standardization activities and hence up to the vendor or the NP, respectively.

1.2 Scientific Contributions

The scientific contributions of this monograph are summarized in Figure 1.2. The research studies are categorized along the horizontal axis according to their focus with respect to the different QoE management building blocks, i.e., *monitoring and measurement*, *modeling and assessment*, or *control and optimization*. The vertical axis denotes whether their relevance is more towards the application domain (or the AP) or towards the network domain (or the NP). The markers $[x]^{y-RQk}$ denote that the scientific publication $[x]$ forms the basis of chapter y and addresses the research question *RQk*. The different chapters are highlighted using different colors.

The first part addresses RQ1 by proposing an analytical approach relying on queueing theory, which is capable of reflecting the behavior of an HAS client. More specifically, the video streaming process is modeled by means of a discrete-time GI/GI/1 system, where the remaining workload in the system represents the buffer filling state of the client. On the one

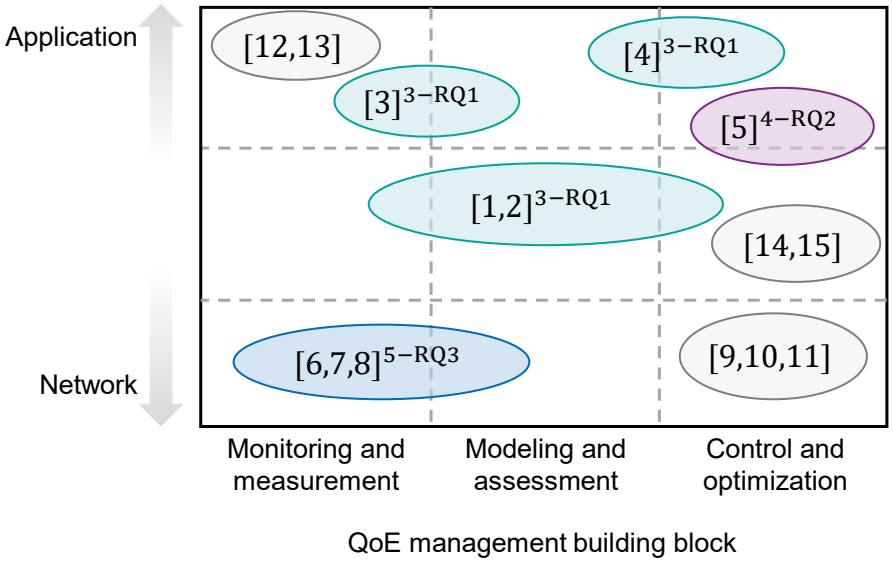


Figure 1.2: Contributions of this work illustrated as a classification of the research studies conducted by the author. The notation $[x]^{y-RQk}$ indicates that the respective publication is discussed in Chapter y of this monograph and addresses research question RQk.

hand, the model is fed with input distributions for the available bandwidth and for the bitrates on the different quality levels. On the other hand, it is supplied with all HAS-specific control variables, e.g., the maximum buffer capacity, the segment duration, and the quality switching thresholds. For the given inputs, the model computes the buffer distribution, from which in turn all QoE-relevant performance metrics can be derived as probabilistic outputs. That is, the stalling probability, the delivered video quality, or the frequency and amplitude of quality switches. The model's analytical nature allows to efficiently explore a huge set of parameter combinations and thus to better understand their interplay within the HAS ecosystem. Despite its high level of abstraction, we show that the model's results are in line with those generated via testbed measurements. Moreover, we demonstrate its applicability for QoE analysis, although it operates on steady-state probabilities, disregarding the temporal behavior of a video session, such as the position of a stalling event during playback. This makes the model a powerful tool to properly adjust the HAS-specific parameters in accordance to each other, so as to optimize the QoE for any given network condition.

Focusing on RQ2, the second part quantifies a possible improvement of the HAS ecosystem by using a content-aware video segmentation strategy. The conventional approach relies on content-agnostic and fixed segment durations, which leads to an increased encoding overhead. This can be eliminated to a certain degree by splitting the video at scene-cuts, which results in variable segment durations. To compare both options from an encoding perspective, we generate a representative data set of video sequences, encoded using a multitude of different parameter combinations. This allows to quantify the encoding efficiency gain, which can be achieved by the proposed approach, as well as to understand the relevant factors, which influence its magnitude. To study the impact of variable segment durations on the streaming performance, we run extensive testbed measurements with varying bandwidth

constraints and different state-of-the-art client-side quality adaptation heuristics. We learn that in certain scenarios, the QoE can be significantly improved without any further adaptations to the HAS ecosystem, while other scenarios would require some adjustments to the system in order to benefit from the variable approach. We elaborate on the root causes of observed drawbacks and indicate how to overcome them via client-side modifications.

The third part targets RQ3 by proposing an integration of Machine Learning (ML) in 5G networks, exploiting the features of its novel network functions, dedicated for the (1) information exchange between the mobile network operator (MNO) and third parties, as well as for (2) enhanced data analytics. We detail on the involved challenges and relevant design criteria when integrating ML-assisted QoE estimation into 5G networks from an operator's point of view. By means of network simulations and on the example of different regression techniques, we practically examine how accurate the QoE can be estimated solely based on network-related KPIs, which are accessible to an MNO. We study the relevance of different types of network telemetry data for the estimation process, as well as possible factors that impact the estimation accuracy, such as user mobility patterns or the used application. Our results can serve as a guideline towards understanding the applicability of different ML techniques, from both, a quantitative and qualitative perspective, as well as to define the relevant statistics and KPIs to monitor.

Finally, we note that for the sake of reproducibility of the conducted studies and to support building upon our obtained results, we publicly provide several data sets and software tools. In particular, the set of encoded video sequences and the streaming sessions obtained via testbed measurements to address RQ2 are provided on Zenodo[1]. The used encoding architecture and the testbed setup are available on Github[2]. The simulation environment which has been implemented in the course of addressing RQ3 can as well be found on Github[3].

1.3 Outline of the Thesis

The structure of this monograph is illustrated in Figure 1.3. The first column denotes the specific goals, the second column represents the applied methodology or the tools that have been used, and the third column specifies the gained insights. Chapter 2 gives general background information on topics related to QoE and HAS. Background and related work which are specific to a certain part of this thesis are introduced at the beginning of the respective chapter. Furthermore, by the end of each chapter, we summarize the lessons learned in a dedicated section.

The focus of Chapter 3 is on the analytical approach for modeling the behavior of an HAS client. We first describe the model's design in its two versions, one representing a buffer-based, the other one a rate-based quality adaptation strategy. Afterwards, we validate the

[1] `https://zenodo.org/record/3732206`
[2] `https://fg-inet.github.io/acm-mmsys-2020/`
[3] `https://github.com/fg-inet/vagrant-omnet-simulation-mobility`

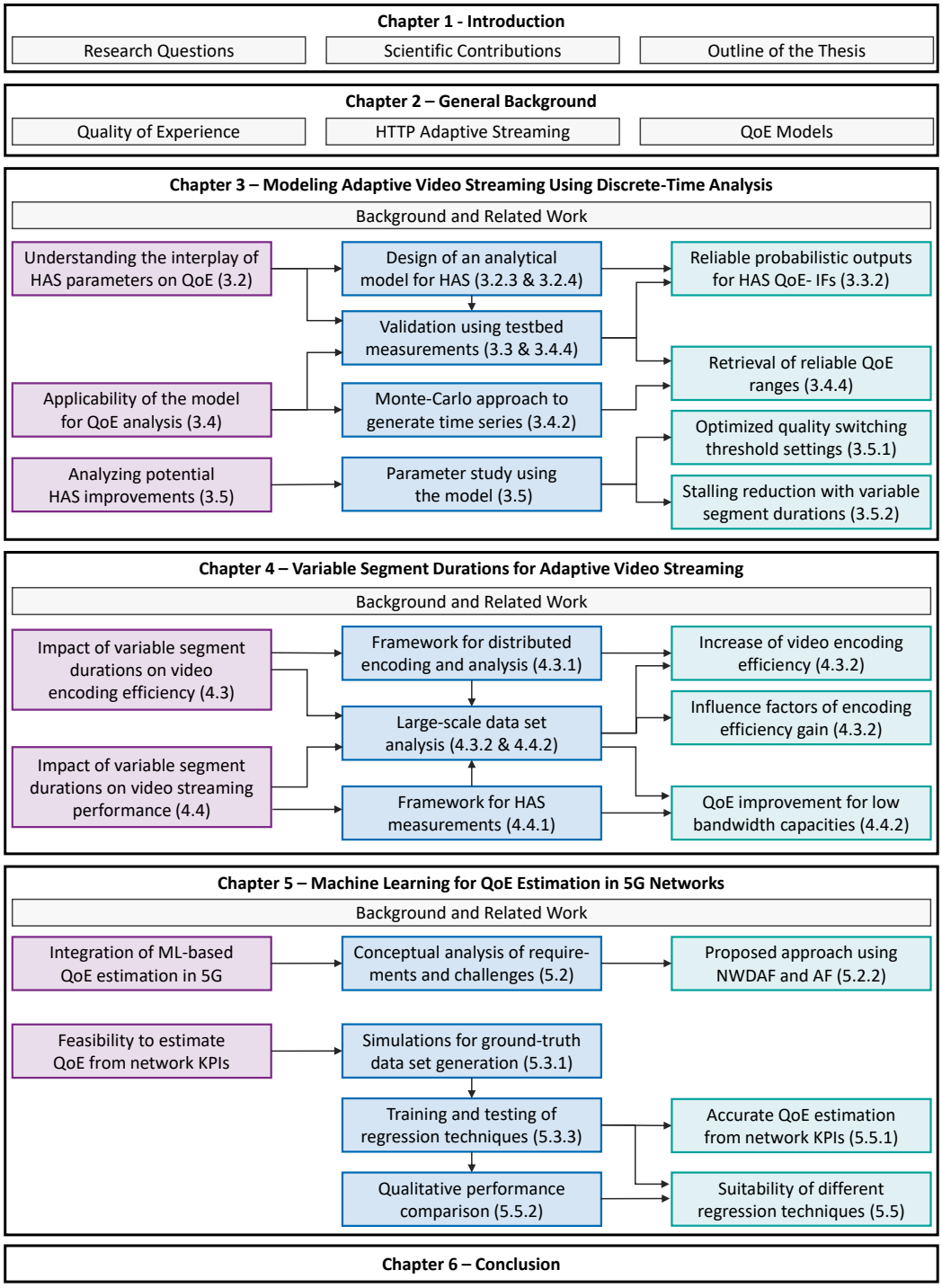


Figure 1.3: Overview of the thesis. The first column (purple boxes) denotes the specific goals. The second column (blue boxes) represents the applied methodology or the tools that have been used. The third column (green boxes) specifies the gained insights. Numbers in brackets denote the respective subsection.

model's capability of computing QoE-related KPIs using testbed measurements. In a next step, we show that in conjunction with a Monte-Carlo simulation approach, the model can be applied for QoE analysis, despite its probabilistic outputs and its missing temporal characteristics. To examine the impact of the quality switching thresholds on relevant performance metrics, we perform an exemplary parameter study using the buffer-based version of the model. Finally, in another experiment, we elaborate on possible improvements of the HAS ecosystem by using segments of variable durations, which reduce the encoding overhead, but introduce more uncertainty to the system.

In Chapter 4 we perform an in-depth analysis of the benefits of variable segment durations. We encode a representative set of videos using various settings for the target quality/bitrate, resolution, and maximum segment duration. The resulting data set contains roughly 2,000 encoded video sequences and allows to holistically compare the variable approach against the state-of-the-art mechanism with respect to several encoding-related parameters, such as the encoding overhead, resulting segment durations, and visual quality. While the variable approach reduces the average bitrate requirements without noticeable quality degradation, it increases the variability of the video segment sizes. To study the impact of this observation on the HAS performance, we perform testbed measurements to create a large data set of streaming sessions. This allows us to compare both approaches with respect to the resulting QoE and to reveal scenarios in which the variable approach is most promising.

Chapter 5 addresses QoE estimation using ML in 5G networks. We first analyze the involved challenges and requirements for realizing such a concept and propose its possible integration into the 5G networking architecture. To study the feasibility of estimating the QoE solely based on network KPIs, we perform network simulations to generate a ground-truth data set, which is used for training ML models. More specifically, we quantitatively compare the performance of five different regression techniques with respect to their estimation accuracy, as well as in terms of their computational overhead during the training and testing phase. Finally, we consider different qualitative aspects which should be kept in mind by an MNO when deciding about the ML model to deploy.

This monograph concludes with Chapter 6 by giving a brief summary on the conducted studies, the obtained results, and the derived insights.

2

General Background

The following chapter describes important concepts covered throughout this monograph. First, we briefly introduce the term QoE and the working principle behind HAS. In this context, we introduce some state-of-the art adaptive bitrate (ABR) strategies and present the HAS control parameter space. Afterwards, we focus on the HAS-related QoE influence factors (QoE-IFs) and present models allowing to retrieve the subjective application quality from objective performance metrics.

2.1 Quality of Experience

Over the past years, the diversity and the demands of Internet applications have been growing. The simultaneous increase of the users' expectations on the quality of the used service [28] made the goal of delivering desirable levels of QoS a highly critical issue [29]. The premise of QoS management is to provide an application with its specific resources, e.g., low delay or packet loss rates for VoIP or high bandwidth for VoD streaming services. Accordingly, QoS research focuses on these network-related KPIs and how they can be measured and evaluated. However, there is a growing consensus in the research community that the technology-centric concept of QoS does not cover sufficiently the performance aspect of a given application or service. QoS neglects the user while the relationship between QoS metrics and subjective user satisfaction is complex [30]. As a consequence, QoS does not allow to infer the user perceived application quality. With the advent of QoE, the term

"'end-to-end quality"' is re-interpreted, considering the user as the end of the communication chain [31, 32]. The Qualinet whitepaper [33] defines QoE as

> *the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and or enjoyment of the application or service in the light of the user's personality and current state.*

The paradigm shift from technical aspects to user perceived quality is beneficial in the sense that users do not bother technical performance, but just expect a website to load quickly or a video to be played back smoothly. Hence, besides the technical factors related to the network, QoE considers several additional factors. Context level factors account, among others, for the user's location or the purpose of using a specific service. User level factors describe psychological effects, such as previous experiences, expectations on the service quality, as well as the time of the day, or the browsing history [34].

Over the past years QoE evolved as an important and broad research topic, covering several aspects. Research activities have been carried out on how to use crowd-sourced measurements to collect subjective user ratings [35, 36, 37]. One of the key challenges thereby is to obtain reliable ratings from the users. This can be hindered by reduced participant investment in the task or by not sufficiently emulating a user's typical environment or platform experience during controlled lab experiments. Several works study the correlation between technical QoS metrics and QoE or QoE-related KPIs for different applications, use-cases, and environments [38, 39, 40, 41, 42]. Other works focus on the impact of different application metrics, such as video interruptions or frame-rate on QoE [43] or elaborate on metrics to quantify QoE [44].

2.2 HTTP Adaptive Streaming

A conceptual overview of HTTP Adaptive Streaming (HAS) is given in Figure 2.1. The video provided at the sever is split into small segments of a few seconds duration [46] and each segment is encoded using different bitrates, resulting in several levels of visual quality. The HAS client first downloads a media presentation description (MPD) file, which contains meta information on the requested video. This includes the video's segment duration, characteristics of the different quality levels, i.e., bitrate and resolution, as well as the URLs to locate the specific segments. The client subsequently downloads the segments via HTTP GET requests and stores them in its video buffer. In order to dynamically adapt the video quality, the client runs an adaptive bitrate (ABR) algorithm, which is often also referred to adapatation heuristic. After a segment download has been completed, the ABR strategy determines the quality for the next segment. Its goal is to maximize the visual quality whilst simultaneously avoiding video interruptions during play back. For instance, it aims at providing the best possible QoE given the underlying networking characteristics.

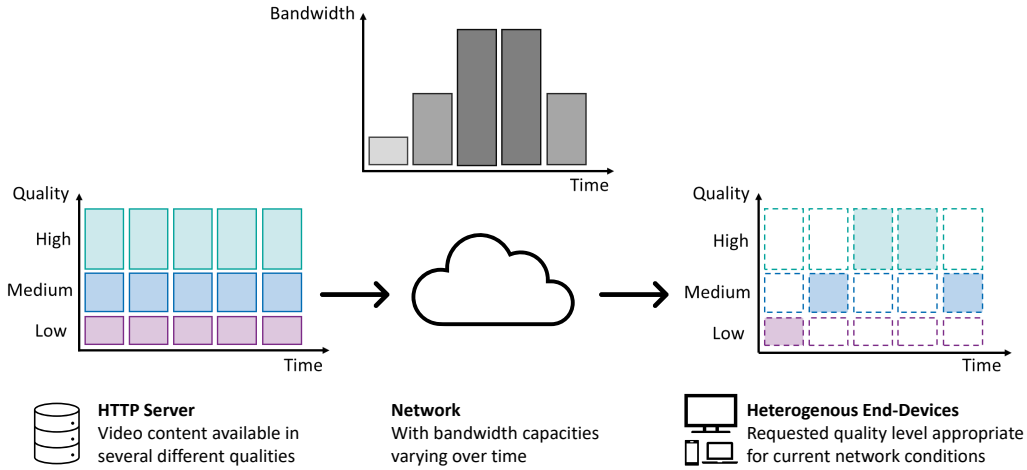


Figure 2.1: Illustration of the working principle behind HTTP Adaptive Streaming [45].

The HAS concept is implemented in different proprietary solutions, such as Adobe HTTP Dynamic Streaming, Apple HTTP Live Streaming (HLS), or Microsoft Smooth Streaming (MSS). Dynamic Adaptive Streaming over HTTP (DASH) is the first streaming solution that has officially been standardized. The DASH Industry Forum (DASH-IF) furthermore publicly provides a reference client implementation, called *dash-js*[1].

2.2.1 Adaptive Bitrate Strategies

Literature propose a vast number of different ABRs [47, 48]. In general, they can be classified as *buffer-based* or *rate-based*. However, some ABRs combine both options and hence can be classified as *hybrid* strategies. Buffer-based approaches adapt the quality depending on the client's video buffer state. A large video buffer allows to request segments of high quality, otherwise the quality is kept low to support a fast download. The quality level is then chosen on buffer thresholds, i.e., the buffer needs to exceed a certain threshold, so to download a specific quality level. ABR mechanisms building on this concept are presented in [49, 50, 51]. One of the most prominent buffer-based ABRs in literature is BOLA [52]. Compared to other buffer-based ABRs, which only consider the number of quality levels to set the quality switching thresholds in a static manner, BOLA tunes these thresholds dynamically based on an optimization function that additionally considers the bitrate of the quality levels.

Rate-based ABR strategies adapt the quality based on throughput estimations. More specifically, the client measures the obtained throughput for each segment download. In order to compensate short-term fluctuations, the measured values are smoothed, e.g., by applying an exponentially weighted moving average. The ABR mechanism then requests the next segment in the highest bitrate not exceeding the measured throughput. Many implementations decrease the measured throughput by a certain factor to obtain a more conservative

[1] https://reference.dashif.org/dash.js/

estimation and to account for overhead. One of the most prominent approaches for rate-based ABR strategies are probe and adapt (PANDA) [53], as well as the strategies which are discussed in [54, 55].

ABR strategies that purely rely on either measured throughput or the current buffer state suffer from certain weaknesses. Throughput estimation techniques applied in rate-based ABRs may not be reliable due to varying bandwidth demands of the client or changing network conditions. Buffer-based adaptation schemes suffer from long-term bandwidth fluctuations [48] and from performance degradations at the beginning of the video playback. As a consequence, there are several proposals to combine rate- and buffer-based approaches, to exploit the benefits of both and overcome the specific drawbacks. FastMPC [56] describes a practical model-predictive controller, which finds an appropriate bitrate for the next segment by combining rate and buffer size predictions. A similar procedure is described in [57], but relying on a fuzzy-based approach.

2.2.2 HAS Control Parameters

By means of a set of parameters, the HAS ecosystem allows to tune the streaming behavior, and thus, the different KPIs. These either relate to client-sided settings or to characteristics of the provided video. In the following, we give a brief overview on these control parameters and how they potentially impact HAS QoE-IFs.

Video segment duration: In general, the video quality can be adapted with each new segment request. Hence, the video segment durations determines a lower bound of how often the quality can be changed throughout the video. The shorter a segment, the quicker the HAS client can react to changing network conditions, while longer segment durations lead to a reduced responsiveness. However, short video segments come with the costs of increased encoding overhead [58] as an IDR frame needs to be placed at the beginning of each segment. This means that for delivering the same visual quality with short segments, a higher bitrate is required compared to using longer segments. Furthermore, as each segment needs to be requested individually, shorter segments increase the overhead in terms of the HTTP GET requests [59]. This is especially problematic in networks with large RTTs.

Number of quality levels: Providing a higher number of quality representations is beneficial for the QoE in two ways. Firstly, it allows a more fine-grained adaptation of the video quality to the current network conditions, and secondly are quality switches between two adjacent levels less recognizable to the user [60, 61]. However, a higher number of quality levels results in increased costs for encoding and storage for the AP.

Buffer threshold settings: Certain thresholds related to the buffer filling state allow to adjust the HAS client's behavior. The first one is the initial buffer state, which determines the minimal amount of buffered play time before the video play back starts. Higher thresholds result in an increased initial delay, but reduce the probability for later re-buffering events. Hence, as the video buffer has significant impact on initial delay and stalling probability,

both being important QoE-IFs, it needs to be carefully dimensioned [62]. The second one is the maximum buffer capacity, defining an upper bound for the amount of pre-buffered video time. While a high maximum buffer supports a fluent play back, it is prone to high bandwidth wastage in case that the user aborts the video stream. Some HAS implementations rely on further buffer thresholds, such as minimum amount of buffered play time after a video stalling, or a video buffer target, which the client aims to achieve.

Quality switching thresholds: Finally, the settings of switching thresholds allow to tune the QoE-IFs. In the case of a buffer-based ABR, the switching thresholds constitute the minimum amount of buffered play time to request a given quality level. Accordingly in the case of a throughput-based ABR, the switching thresholds constitute the minimum measured throughput to download a specific quality level. High, i.e., pessimistic, settings of the switching thresholds result in lower visual quality but simultaneously reduce the risk of stallings, while optimistic settings deliver higher quality at an increased stalling risk.

Finding an optimized tuning of these parameters is a challenging task, as the various operational settings need to be geared to each other. Furthermore are there several external influencing factors like the client's device screen size or the underlying network characteristics. For example, in mobile networks with higher bandwidth fluctuations, thresholds for quality switches and buffer settings should be chosen more pessimistic. There are several proposals to tackle the issue of dynamically tuning the parameter space to the given conditions. The Adaptation Buffer Management Algorithm (ABMA) [63, 64] dynamically adjusts the video buffer size to compensate for download rate oscillations. The Buffer Occupancy based Lyapunov Algorithm (BOLA) [52] adjusts the quality switching thresholds to the bitrate characteristics of the available quality levels. Similar to that, SARA [65] takes the variability of the video segment sizes into account when selecting the next quality.

2.2.3 QoE Influence Factors for HAS

The QoE for HAS is impacted by different factors related to waiting times and quality [66]. In the following, we give a brief introduction on all QoE-relevant KPIs for HAS and how they affect the resulting user QoE.

Initial delay: The initial delay describes the user's waiting time between initiating the video stream and the video play back. Users are more sensitive towards waiting times during the play back and rather accept a longer initial waiting time [67]. Consequently, in order to avoid interruptions during the play back, typical HAS implementations define a threshold for the minimum amount of buffered play time, before the video is played back. To avoid QoE degradation, a good trade-off needs to be found which keeps the initial delay as low as possible, whilst still allowing a fluent play back once it started.

Stallings: This term describes interruption during the video play back, resulting in waiting times for the user. The video typically stalls due to a re-buffer event, i.e. the buffer drained empty and the play back can only be resumed, when a certain buffer threshold is reached

again and earliest as soon as the next segment is arrived. With respect to video stallings, there are two KPIs influencing the QoE: the stalling duration and the overall number of stallings occurring throughout the video play back. Studies show that the user's QoE is degraded to higher extent, if more interruptions of shorter duration occur, compared to a reduced number of stallings of longer duration [68].

Quality switches: The adaptive behavior of HAS results in changing quality throughout the video play back, causing viewer distraction. There are two important KPIs related to the quality switches. Firstly, the frequency or overall number of switches, and secondly, the switching amplitude [60, 69]. In order to provide a good QoE, a solid trade-off between maximizing the delivered video quality and minimizing the frequency and amplitude of quality switches needs to be found [70].

Visual quality: The resulting visual quality of a video is determined by the used encoding bitrate, the video's resolution, and the complexity of the video content. Metrics for assessing the visual quality can be classified into three major categories [71]: full-reference (FR), no-reference (NR), and reduced-reference (RR). FR metrics perform a frame by frame comparison between a reference video (usually uncompressed and hence without quality degradation) and the video under test. The two most prominent FR metrics are the Structural Similarity Index Measure (SSIM) [72] index and the Peak Signal to Noise Ration (PSNR) [73]. The Fusion-based Video Quality Assessment (FVQA) [74] aims at assessing the visual quality of a video stream more reliably, by fusing several existing video quality assessment methods to a single video quality score. Thereby, the fusion coefficients are learned from training video samples. This approach was then further developed in conjunction with Netflix and is today known as the Video Multimethod Assessment Fusion (VMAF), which constitutes an approach widely adopted by research and industry [75]. NR metrics assess the visual quality without an explicit reference video. Most of the NR metrics rely on blockiness [76, 77], which is the prominent artifact of block-based compression techniques such as H.264. RR metrics combine the principles of FR and NR metrics. Hence, RR metrics combine the benefits of high accuracy of the quality assessment by using a reference sequence and of a reduced number of features for comparing the reference video against the video under test [78].

2.3 QoE Models

Explicit user ratings, e.g., in the form of feedback questionnaires, are the most reliable measure for the customer satisfaction. However, they are practically not usable on large scale as the participation is typically limited and surveys are prone to disturb the user [79]. QoE models allow to derive the perceived quality based on application- and/or network-related KPIs. Typically, user studies are performed in the scope controlled lab studies, where the test subject is exposed to different stimuli, e.g., a specific video quality or stalling pattern, and then asked to rate the level of satisfaction on a pre-defined scale. The average of all test subjects' scores is often referred to as the mean opinion score (MOS) [80], which evolved

over time as the de-facto metric to quantify perceived media quality [81]. In particular, the 5-point MOS scale is very popular. It ranges from 1 to 5 and represents the perceived quality as bad, poor, fair, good, and excellent. Correlating the network- and application specific settings with the users' ratings allow to mathematically model the impact of different stimuli or network characteristics on the resulting user satisfaction. Hence QoE models allow to estimate the users' perceived quality also in the absence of explicit ratings. In the following, we present the two QoE models for HAS applied in this work and give a brief outline on further models.

2.3.1 Standardized ITU-T P.1203 Model

The ITU-T recommendation P.1203 describes the first standardized QoE model for HAS [82, 83]. Its high accuracy in terms of predicting the real streaming QoE could be obtained via extensive training and validation on a huge database, containing over a thousand video sequences. These video sequences contain HAS-typical artifacts, such as stallings, quality switches, initial loading time, or visual quality degradation resulting from low encoding bitrates, as well as the subjective rating, expressed as MOS. The model consists of three *modules*, one for the visual video quality (P_v), one for the audio quality (P_a), and finally one for the audio-visual integration (P_q). In the remainder of this work, we will confine on the visual quality (P_v) and omit the audio quality score. Furthermore can the video quality itself be estimated in three different *modes*. Thereby, lower modes require less information at the cost of lower accuracy and can be computed more efficiently. *Mode 0* estimates the QoE only based on meta-data. For instance, initial loading times, stalling events, resolution, bitrate, framerate, and the used video codec. *Mode 1* extends the included information by packet header information, such as the video frame size and whether a frame is an I-frame or not. *Mode 2* and *mode 3* additionally consider video bitstream information, up to 2% in *mode 2* and full bitstream parsing in *mode 3*. This mode hence uses video frame-level characteristics, such as the frame types, theirs sizes and the quantization parameter (QP) values on a per-macroblock scale. Thus, *mode 3* yields the highest QoE estimation accuracy that is possible with the P.1203 model. The model returns one overall quality score and several diagnostic quality scores on a MOS scale. We will use the following scores throughout the scope of the monograph:

- **O34:** Per-second audiovisual quality score
- **O23:** Stalling quality
- **O46:** Overall quality score, combines audiovisual and stalling quality scores

When omitting the audio track, the model per default assumes a constant high audio quality when computing the audio-visual quality score (O34). Furthermore, O34 yields a value for each second of the video stream. When we refer to O34 in later parts of this work, we mean the average of all per-second scores of a streaming session.

An in-depth study of the impact of HAS QoE factors on the output of P.1203 is performed in [84], showing that well-known results from previous QoE studies are qualitatively represented, such as the high impact of stallings. As P.1203 has been trained and validated only for H.264 encoded video sequences and resolutions up to full-HD, [85] provides an extension towards newer codecs like H.265 and VP9, as well as resolutions up to 2160p.

2.3.2 Cumulative Quality Model

Another model to asses QoE for HAS is the Cumulative Quality Model (CQM) presented in [86, 87]. Compared to ITU-T P.1203, which retrieves one QoE score for the whole video session, CQM evaluates the quality of a session over time, making the model also applicable for real-time quality monitoring. The model relies on time windows of different lengths, so to account for short-term quality fluctuations, e.g., between subsequent segments, as well as for long-term quality fluctuations, to compute the cumulative quality at any point in time. For a given window, each contained video segment's visual quality is assessed via existing metrics [88, 89]. Via a weighted sum of the visual qualities, interruption patterns, and quality switches, the per-window quality value is determined. For the first window, the initial delay is considered additionally. Please note that the output of CQM can be influenced by different parameter settings, i.e., it allows to weight the QoE-IFs differently. In this work, we use the implementation which is publicly provided[2] along with the default weight settings.

2.3.3 Other QoE Models

A simple QoE model for HAS only taking the time on highest layer and the quality switching amplitude into account is presented in [60]. The work in [56] assesses QoE as a weighted sum of average video quality, the variation of the quality between the segments, stalling events, and the initial delay. Yet another approach is presented in [90]. Besides the typical QoE-IFs, the proposed method additional considers the bitrate distribution and takes into account the recency effect, a psychological phenomenon describing that users tend to remember recent events better [91]. QoE models exist for a wide range of different applications. For example, the work in [92] allows to retrieve the QoE for file download applications, by deriving the MOS as a logarithmic function from the overall download duration. Remote virtual desktop services are targeted in [93]. The authors evaluate how the MOS score of typical tasks, such as typing, scrolling, or menu browsing, is affected by the RTT or by the available bandwidth. The work in [94] studies how framerate and bitrate affect the cloud gaming quality and proposes a QoE model based on these two effects for two exemplary games.

[2] `https://github.com/TranHuyen1191/CQM`

3

Modeling Adaptive Streaming Using Discrete-Time Analysis

As outlined in the previous chapter, the overall HAS performance is influenced by a variety of factors, including the applied ABR scheme, the underlying network characteristics, as well as video- and player-specific parameter settings. These influencing parameters are, upon others, quality switching thresholds, upper and lower buffer bounds, segment durations, the average video bitrate and its variation, or the number of provided quality levels. In order to tune these parameters for optimizing the QoE, both, testbed measurements and simulations, have been widely adopted [53, 95]. However, the large number of adjustable and non-adjustable parameters, with each of them theoretically having an unlimited value space, results in a vast number of potential parameter combinations. As a consequence, neither measurements nor simulation activities scale to study the complete parameter space. Hence, current evaluations are performed only for specific use cases and are only capable of covering a subset of the large problem space.

As no holistic evaluation methodologies exist so far, we only have a very limited understanding of the effects that the various parameters have on the performance metrics across the wide range of operational settings. In the following, parameters refer to the factors such as the number of quality levels, the quality switching thresholds, or the segment durations, while the operational settings refer to external influencing factors like network characteristics or the video content. Therefore, even for a single operational setting, it remains unclear

which parameter configuration would yield the best performance. Furthermore, it is practically infeasible to identify *relevant* factors and inter-dependencies between these, which is required for a comprehensive understanding of the factors significantly influencing the performance of adaptive video streaming.

To tackle the problem of a holistic understanding, different analytical queueing-based approaches have been developed [62, 96]. Even though both models allow for a fast computation of a subset of relevant HAS performance metrics, such as the stalling probability or the stalling duration, they do not allow to cover all of the previously outlined influence factors. This is due to their strong assumptions, upon others that the quality is not dynamically adapted to the underlying network conditions or the buffer state. Consequently, they fail to analyze any crucial metric related to the playback quality, such as the frequency or amplitude of switches, or the average visual quality.

To enable the incorporation of these HAS-inherent parameters, this chapter proposes a queueing model based on discrete-time analysis, which is capable of taking the adaptive behavior of an HAS client into account. More precisely, we extend the work from [96] by the inclusion of multiple video quality representations and by the integration of the decision logic behind rate-based and buffer-based ABR strategies. The proposed model overcomes the limitations of current solutions and thus allows for analyzing the full range of HAS-related operational settings, as well as their interplay and implications in terms of QoE-relevant performance metrics. We confirm this ability by providing a testbed-driven validation of the model, using a vast number of parameter combinations and varying available bandwidths settings.

The model returns distribution functions, e.g., for the stalling probabilities, without any information about the temporal behavior of a video session, such as the timely positions of stallings. It hence needs to be examined whether QoE prediction models, such as the P.1203 model, which compute the QoE based on the chronological sequence of a specific video playback, can be used when the QoE-relevant metrics are obtained from the HAS performance model, i.e., without any temporal information. For that reason, this chapter additionally evaluates how and to which extent the generalized results of the analytical model can be utilized to derive sequence-based QoE values or the QoE distribution for a set of sequences for similar input parameters with stochastic variations. To do so, we derive chronological video playback sequences using a Monte-Carlo approach for a specific video clip and different network scenarios. For these sequences, we compute the perceived QoE distributions using the open source implementation of P.1203. Similarly, we conduct testbed measurements for the same network scenarios using the Bola [95] ABR controller implemented in the dash.js framework and further compute the corresponding QoE distribution. Our evaluations show that using the output of the HAS performance model to generate artificial video sequences which are used along with the P.1203 QoE prediction model allows for a good QoE estimation.

Finally, this chapter shows two exemplary use-cases for a practical application of the buffer-based version of proposed model. The first one is a small parameter study, which evaluates

the impact of the switching threshold setting on different HAS-relevant QoE-IFs. Our analysis shows that there are indeed configurations, which are beneficial for all tested network scenarios. The second one evaluates potential prospects of using a different video segmentation technique, which relies on variable instead of fixed-length segments. More specifically, we show that by the variable approach, a significant reduction of stalling probability can be achieved.

The remainder of this chapter is structured as follows. Section 3.1 provides background information on analytical models and presents related work which are using such models for HAS analysis. Section 3.2 focuses on our proposal of an analytical model which is based on discrete-time analysis. Thereby, we first introduce the used notations and describe the stochastic preliminaries. After that, both, the buffer-based and the rate-based version of the model are introduced. In a next step, we show how QoE-related video streaming metrics, such as average quality or stallings, can be derived from the modeled video buffer. In Section 3.3 we validate the model's accuracy by comparing it against video traces obtained via testbed measurements. Subsequently, we investigate in Section 3.4 in how far the model can be applied for QoE analysis, despite the unavailability of temporal information of the video session. Two use-cases for applying the model are presented in Section 3.5. The first one presented in 3.5.1 is the exemplary parameter study on the quality switching thresholds, while the second one evaluates in 3.5.2 the possible benefits of using variable durations for video segments, instead of relying on fixed ones. Finally, we summarize the lessons learned in Section 3.6.

3.1 Background and Related Work

Theoretical models allow to analyze the processes of (complex) systems in an abstracted manner. Based on the works from Tran-Gia and Hoßfeld [97, 98], the following Section first introduces the key working principle of queueing-based models and presents two generic discrete-time approaches afterwards, which form the basis for the HAS performance model presented later in this chapter. Several generic HAS models have been presented in the literature and we outline the respective relevant publications. Moreover have analytical models been considered for an integration into ABR strategies in order to optimize their decisions. Hence, the following Section briefly outlines research activities towards improving ABR mechanisms by means of analytical models.

3.1.1 Analytical Models and Queueing Theory

Analytical models allow to theoretically describe a system's processes, taking into account the most relevant timely and logical characteristics. They are often used to assess the performance of a system when being under load, to retrieve rejection probabilities or to obtain the waiting or processing times. If the number of participants or requests in such a system is sufficiently large, the system's procedures can be modeled by means of stochastic

processes [98]. While actual measurements yield the best reflection of a real system, simulations introduce abstractions of the system to different extents, depending on the level of detail with which the system components are simulated. In this context, analytical models, which rely on approximative methods, have the highest level of abstraction. However, such a highly abstracted approach which only considers the most essential system properties allows to explore a huge parameter space very efficiently. Due to their high efficiency, analytical models are widely used for example for determining how to scale a system so to satisfy certain performance requirements.

Analytical models relying on queuing theory basically analyze arrivals and processing times to identify flaws of the system and to retrieve optimized scaling of resources. Thereby, the modeled system typically includes a source, one more processing units and optionally one or several queues, which can be finite of infinite. For example, one could imagine modeling the payment process of customers in a supermarket. The customers arrive at the checkout with a specific rate from which their inter-arrival time can be deduced, i.e., the time that passes between two customers being ready to pay their groceries. This can be seen as request, which should be served by one of the processing units, i.e., the checkstand. If at least one checkstand is currently not serving any other customer, the request can directly be served. If all checkstands (processing units) are blocked, the customer has to wait. There can either be one global queue, or one queue for each checkstand. In any case, the customer's waiting time depends on the following two factors: The number of earlier arrived customers waiting in line, and the duration of scanning their products and cashing them up, which is referred to as the processing time. For a given inter-arrival time of customers and a given processing time, which are expressed as random variable, the queuing-based model allows to compute the waiting time of customers, depending on the number of opened checkstands and the queue management (one vs. several queues) and can hence be used to optimize the system. Systems where an endless number of requests can be queued are referred to non-blocking systems. In contrast, blocking systems reject requests either when the queue capacity is reached, or when there is no queue and all serving units are blocked. A typical example for a blocking system is a mobile cell with a limited number of communication channels, which are assigned to the subscribers connected to the cell. In case that all of these channels are already allocated, any new subscriber is blocked from the system. While in non-blocking systems, the focus is typically to minimize waiting times, queuing-based models for blocking systems can be used to scale the system so to not exceed a certain blocking rate with a given probability.

How exactly a system can be modeled depends, besides being non-blocking or blocking, on different other characteristics. This includes upon others the stochastic properties of the inter-arrival and processing times, i.e., their specific distributions, as well as the number of processing units and the number of waiting slots, i.e., the queue capacity. Furthermore can queueing-based models either be discrete or continuous in terms of time and state.

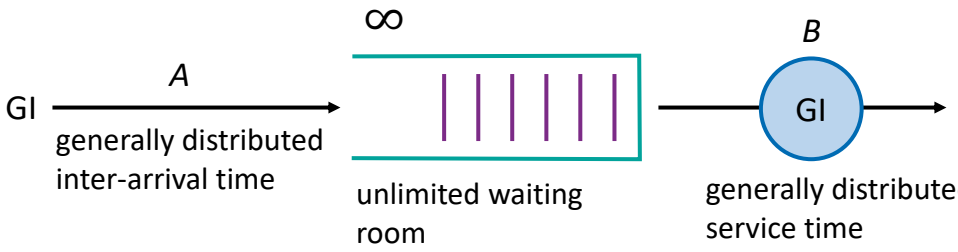


Figure 3.1: Schematic illustration of the discrete-time delay system GI/GI/1 [97].

3.1.2 Discrete-time GI/GI/1 Queueing Models

Discrete-time models operate on discrete time intervals of constant length Δt. The system is then only analyzed for the resulting set of equidistant points in time. The notion GI/GI/1 indicates that the distribution for both, the inter-arrival time A as well as the service time B, is general independent (GI) with:

$$a(k) = P(A = k \cdot \Delta t) \quad k = 0,1,2,...,$$

$$b(k) = P(B = k \cdot \Delta t) \quad k = 0,1,2,....$$

An illustration of a GI/GI/1 discrete-time delay system is shown in Figure 3.1. It considers a single service unit and one queue with infinite capacity. Typically, GI/GI/1 systems are analyzed with respect to the amount of unfinished work in the system, denoted as $U(t)$. We use Figure 3.2 to show how $U(t)$ can be derived from the Random Variables (RVs) A and B and introduce the following notations:

A_n RV for the inter-arrival time between request n and request $n+1$

B_n RV for the service time of the n-th request

U_n^- RV for the amount of unfinished work in the system immediately before the arrival of the n-th request

U_n^+ RV for the amount of unfinished work in the system immediately after the arrival of the n-th request

U_{n+1}^v RV for the virtual amount of unfinished work in the system immediately before the arrival of the request $n+1$.

The x-axis denotes the time t, while the y-axis represents the amount of unfinished work in the system at a point in time t, i.e., $U(t)$. In general, $U(t)$ corresponds the sum of service times of all waiting requests plus the remaining service time of the request currently being processed by the service unit. Accordingly, $U(t)$ increases with each incoming request n by the service time of the n-th request, denoted as B_n. For the arrival of new request n, two distinct RVs are considered. Firstly, U_n^-, which denotes the amount of unfinished work *immediately before the arrival* of n. Secondly, U_n^+, which respectively denotes the amount

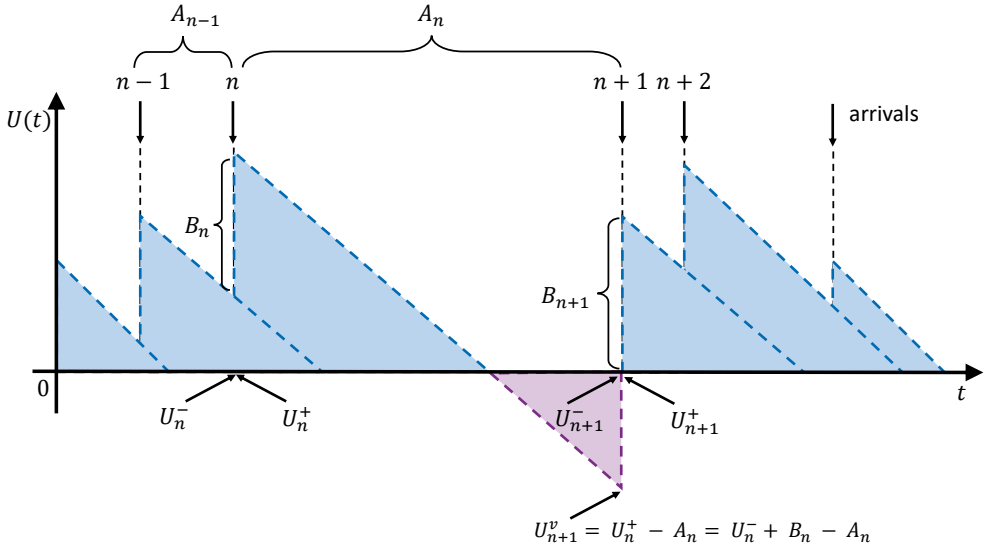


Figure 3.2: Course of request arrivals and evolvement of the unfinished workload in a GI/GI/1 system [97].

of unfinished work in the system *immediately after the arrival* of n. Another additional RV used is the virtual buffer U^v, which can be negative and is introduced to simplify computations. It is capable to account for the time during which the service unit is empty.

The amount of unfinished work in the system directly prior to the arrival of the $n+1$-th request (U_{n+1}^-) is computed from the amount of unfinished work prior to the arrival of the n-th request (U_n^-) as follows: First, the service time of request n is added to U_n^-:

$$U_n^+ = U_n^- + B_n$$

In general, considering two random variables F and G with density functions f, g we note that the sum $H = F + G$ is distributed according to the density function $h = f * g$, where $*$ denotes the convolution of density functions, *when* F and G are independent. While the convolution inherently is an operation on functions and again yields a function, the notation $h(k) = f(k) * g(k)$ is often used in the queuing theory literature to denote $h = f * g$ (see further [96, 99]) and we adapt our notation accordingly. An advantage of this notation is the simpler expression when subtracting two random variables: $H = F - G$ is then denoted by $h(k) = f(k) * g(-k)$, where $g(-k)$ denotes the probability density function $-k \mapsto g(k)$.

Accordingly, under the assumption that U_n^- and B_n are independent RVs with the respective density functions u_n^- and b_n, the density of the unfinished work directly after arrival of request n, is denoted as

$$u_n^+(k) = u_n^-(k) * b_n(k). \tag{3.1}$$

The next step computes the virtual buffer immediately before the arrival of the $n+1$-th

request, denoted as U^v_{n+1}. For that, the inter-arrival time of $n+1$ is subtracted from U^+_n:

$$U^v_{n+1} = U^+_n - A_n$$

Denoting $U - A$ as $u(k) * a(-k)$, the density function for U^v_{n+1} is derived as follows:

$$u^v_{n+1}(k) = u^+_n(k) * a_n(-k) \tag{3.2}$$

The virtual amount of unfinished work in the system differs from the actual amount of unfinished work only in the sense, that the virtual can have negative values, while the actual cannot. Accordingly, for U^-_{n+1} it holds:

$$U^-_{n+1} = max(0, U^v_{n+1})$$

To obtain the respective density function, all probability mass below 0 of u^v_{n+1} has to be shifted to 0, which is achieved with the π_0 operator:

$$u^-_{n+1}(k) = \pi_0\big(u^v_{n+1}(k)\big) \tag{3.3}$$

Whereby the π_0 operator applied here is a special case for the generic discrete operator π_m, which is defined as follows:

$$\pi_m\big(x(k)\big) = \begin{cases} 0, & k < m, \\ \sum_{i=-\infty}^{m} x(i), & k = m, \\ x(k), & k > m. \end{cases} \tag{3.4}$$

Combining Equations 3.1, 3.2, and 3.3, the density function for the remaining work load in the system immediately prior the arrival of request $n+1$ can be condensed to:

$$\begin{aligned} u^-_{n+1} &= \pi_0\left(u^-_n(k) * b_n(k) * a_n(-k)\right) \\ &= \pi_0\left(u^-_n(k) * c_n(k)\right) \end{aligned}$$

Thereby,

$$c_n(k) = b_n(k) * a_n(-k)$$

is called the discrete-time system function.

If the service unit processes the requests in the queue in a first-in first-out (FIFO) manner, the waiting time of request n corresponds to the amount of unfinished work immediately before its arrival, i.e., U^-_n. Accordingly, the waiting time of the n-th request, denoted as RV

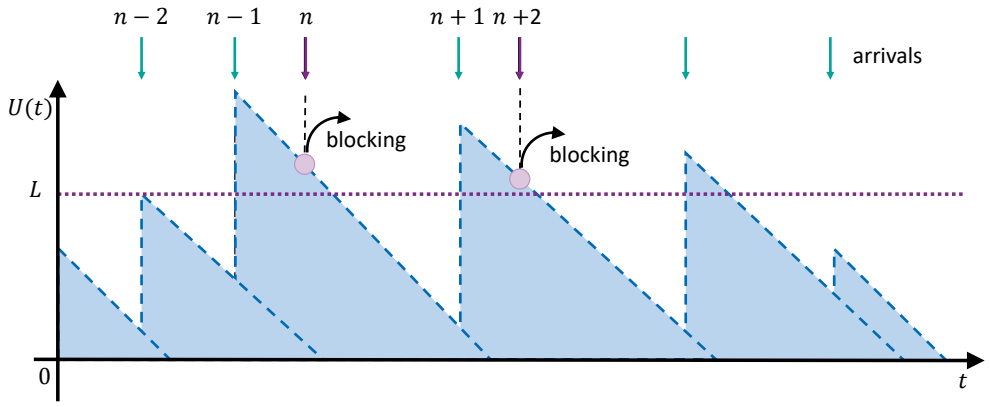


Figure 3.3: Discrete-time delay system GI/GI/1 with bounded delay [97]. The requests n and $n+2$ are blocked as the remaining workload in the system at their arrival exceeds threshold L.

W_n, can recursively be computed as follows:

$$W_{n+1} = max(0, W_n + B - A)$$
$$w_{n+1} = \pi_0\big(w_n(k) * c_n(k)\big)$$

Under the assumption that inter-arrival times and service times are statistically independent and identically distributed, i.e., $A_n = A$ and $B_n = B$ for all n, and that the system is stationary ($W = \lim_{n\to\infty} W_n$), the density for the waiting time can be denoted as

$$w(k) = \pi_0\big(w(k) * c(k)\big)$$

with

$$c(k) = b(k) * a(-k).$$

3.1.3 GI/GI/1 System with Bounded Delay

Queues with infinite capacity can result in huge waiting times. In some cases, it is, however, desirable to bound the waiting time for incoming requests to an upper limit. This can be achieved by rejecting an incoming request if its expected waiting time exceeds a pre-defined threshold. The system introduced in 3.1.2 is modified in such a way, that only those requests are accepted and enqueued, which have at the point of their arrival an expected waiting time below

$$W_{max} = L \cdot \Delta t.$$

Such a system is illustrated in Figure 3.3. The request is not served, if the remaining workload exceeds L, i.e., if $U_n \geq L$. Hence, for GI/GI/1 systems with bounded delay, two cases need to be considered. In the first case, the request is accepted as the threshold L is not

reached. The remaining workload in the system is computed using the conditional RV $U_{n,0} = U_n | U_n < L$ as follows:

$$u_{n,0}(k) = \frac{\sigma^{L-1}[u_n(k)]}{P(U_n < L)} = \frac{\sigma^{L-1}[u_n(k)]}{\sum_{i=0}^{L-1} u_n(i)}.$$

The operator $\sigma^m[x(k)]$ only accepts the lower part ($k < m$) of a distribution and leaves out the rest:

$$\sigma^m[x(k)] = \begin{cases} x(k), & k \leq m, \\ 0, & k > m. \end{cases} \tag{3.5}$$

Through the division with $P(U_n < L)$ the distribution of RV $U_{n,0} = U_n | U_n < L$ is normed. The amount of unfinished work in the system in case the request is accepeted evolves as follows:

$$\begin{aligned} U_{n+1,0} &= U_{n,0} + B_n - A_n, \\ u_{n+1,0} &= \pi_0[u_{n,0} * b(k) * a_n(-k)] \end{aligned}$$

In the second case the request is rejected because the threshold L is reached. The distribution of the conditional RV $U_{n,1} = U_n | U_n \geq L$ is defined as follows:

$$u_{n,1}(k) = \frac{\sigma_L[u_n(k)]}{P(U_n \geq L)} = \frac{\sigma_L[u_n(k)]}{\sum_{i=L}^{\infty} u_n(i)}.$$

The operator $\sigma_m[x(k)]$ only accepts the upper part ($k \geq m$) of a distribution $x(k)$ and leaves out the rest:

$$\sigma_m[x(k)] = \begin{cases} 0, & k < m, \\ x(k), & k \geq m. \end{cases} \tag{3.6}$$

Similar as above, the division with $P(U_n \geq L)$ norms the distribution of the RV $U_{n,1} = U_n | U_n \geq L$. The amount of unfinished work in the system in case the request is rejected evolves as follows:

$$\begin{aligned} U_{n+1,1} &= U_{n,1} - A_n, \\ u_{n+1,1} &= \pi_0[u_{n,1} * a_n(-k)] \end{aligned}$$

When combining both cases - accepting and rejecting the next request - the unfinished work at arrival of $n+1$ can be derived from U_n as follows:

$$u_{n+1}(k) = P(U_n < L) \cdot u_{n+1,0}(k) + P(U_n \geq L) \cdot u_{n+1,1}(k)$$

Using this equation allows to establish a recursive formula to obtain the remaining workload

in the system immediately before arrival of request $n+1$:

$$\begin{aligned} u_{n+1}(k) &= \pi_0\left[\sigma^{L-1}[u_n(k)] * b_n(k) * a_n(-k)\right] + \pi_0\left[\sigma_L[u_n(k)] * a_n(-k)\right] \\ &= \pi_0\left[\sigma^{L-1}[u_n(k)] * b_n(k) * a_n(-k) + \sigma_L[u_n(k)] * a_n(-k)\right] \end{aligned}$$

For a stationary system, the probability of being blocked corresponds to the probability of the amount of unfinished work being above L and hence:

$$p_B = \sum_{k=L}^{\infty} u(k).$$

3.1.4 Generic HAS Models

The HAS behavior can be modeled by using Markov models. One example are $M/M/1/\infty$ models. They work on a high level of abstraction, but in return they allow to easily compute relevant metrics. Hoßfeld et al. [62] presents such a model, which applies a pq-policy. Thereby, buffer values of p and q constitute lower and upper buffer bounds for segment requests. For instance, as long as the video buffer is below p seconds, segments are requested to ramp up the buffer. As soon as the upper threshold of q seconds is exceeded, the client enters the idle state and does not request further video segments, until the buffer falls below the threshold p again. This results in the HAS-typical on-off behavior and using mean-values analysis, the authors are capable of deriving stalling frequency and stalling ratio. The model and its capabilities for computing these metrics is then applied to optimally dimension the video buffer for users with different characteristics, e.g., watching a complete video versus video browsing.

De Cicco et al. [100] formalizes the behavior of an Akamai video streaming session. The system is modeled as a hybrid automaton, using upon others the video level, the current rate, and the playout buffer as state variables. Using their model, the authors show that stalling can be avoided by properly tuning switching thresholds and that a proper setting of the ratio between idle states and segment downloading can avoid large buffering, which results in network resource wasting in case the user aborts the video.

Burger et al. [96] models the video buffer as a $GI/GI/1$ queue with pq-policy using discrete-time analysis. Thereby, the video portion buffered at the client is considered as the amount of unfinished work in the system. The playback corresponds the service time, i.e., draining the buffer. It is assumed that the inter-arrival times, which correspond to the segments' download durations, are independent. The model allows to evaluate the impact of video characteristics (e.g. segment duration, bitrate variation), network dynamics, and buffer policies on the streaming performance. However, this work does not model the quality switching behavior of HAS. Admittedly, the model allows for evaluating metrics like stalling probability and average buffer, but not for evaluating those QoE influencing factors that are related to video quality, i.e., the average quality or the number and amplitudes of quality switches. Correspondingly, the model in it its current state does not yet allow to examine

the impact of the number of quality levels, or the setting of quality switching thresholds, on HAS performance.

A queueing theoretic method to predict video interruptions in wireless environments is presented in [101]. It models the client's video buffer as a G/G/1 queue. Building on the previous work from [102], the authors propose a method that is capable of predicting suitable video streaming parameters, e.g., the initial buffer filling threshold, for given network conditions, such as the delay. As a novelty compared to previous works [103, 104], the authors incorporate the effects of finite video durations into their buffer model. Another video buffer model for wireless networks is presented in [105]. Compared to the previously discussed approach proposed in [101], this model relies on a G/D/1 queue. For instance, while the packet arrival is still arbitrary, the video play back is assumed to be deterministic.

3.1.5 HAS Models for ABR Improvement

Several works model the behavior of adaptive video streaming in the scope of developing more advanced ABR algorithms. One of them is QUETRA [106], which makes use of an $M/D/1/K$ queue to model the client's buffer taking into account the chosen video segment quality, current network throughput, and the overall buffer capacity limit. Hence, their model allows to estimate the buffer filling state depending on potential bitrate decisions and the current network state. This is exploited to optimize the client's decision in terms of the next segment to download and consequently to obtain an ideal buffer filling state.

Another proposal, also relying on an $M/D/1/K$, is qMDP [107]. While QUETRA considers an arbitrary buffer capacity, qMDP aims at dynamically optimizing it via reinforcement learning (RL). Therefore, they introduce rewards for training the RL-based model. Positive rewards are given for downloading a high video bitrate, while negative rewards are given for quality switches, stalling events, and deviations of the current buffer from the targeted buffer value.

A similar approach, called DQ-DASH [108], makes use of an $M^x/D/1/K$ queueing approach. Compared to QUETRA and qMDP, DQ-DASH considers parallel video segment downloads from multiple servers. For that reason, the segment download is modeled as Markovian batch arrival process. The proposed model is incorporated into an ABR strategy, which - besides the buffer estimated - takes the throuhput measurements to the different servers as well as the total buffer capacity limit into account when decding about the next segments' quality level.

Model predictive control (MPC) is used in [56] to optimally combine rate-based and buffer-based adaptation heuristics. In general, MPC attempts to predict key environment variables over a moving look-ahead horizon and to solve an exact optimization problem based on the prediction. To overcome the high computational overhead and practical difficulties, such as additional software requirements in the video player, the authors propose FastMPC. It stores optimal control decisions offline, so to use them later efficiently online.

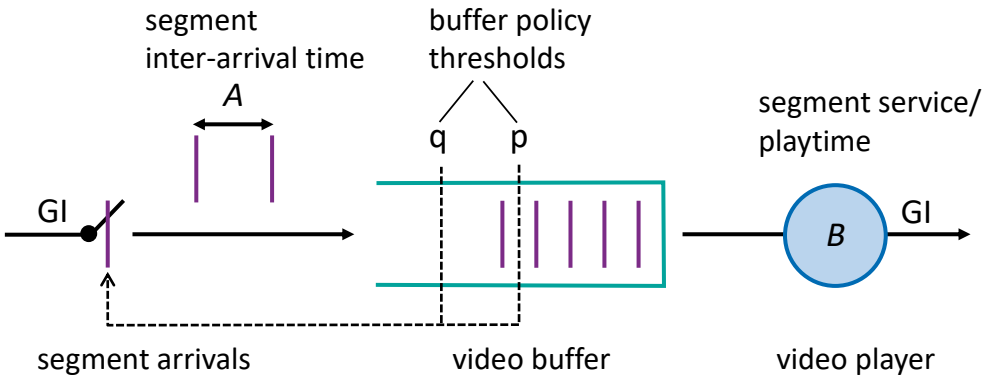


Figure 3.4: Discrete-time delay system GI/GI/1 for video buffer modeling [96].

3.2 Proposed Discrete-Time Models for HAS

This section details on the proposed model for HAS, which is capable to model the quality switching behavior. It builds upon the previous work [96] and extents it in such a way, that either from the buffer distribution or from the throughput distribution, a distribution for the quality levels can be derived. We first give a short overview on the basic model, introduce the notations used throughout this chapter, and present the stochastic preliminaries. Then, the both versions of the model, one reflecting a buffer-based and one reflecting a rate-based adaptation strategy, are described. In a next step, we show how the relevant metrics can be obtained from the modeled video buffer for both of the versions. Finally, we show the validation results.

3.2.1 Model Overview and Notations

A systematic overview of the model is given Figure 3.4. It is an adaptation and specification of the system denoted in Figure 3.1 towards the use-case of video buffer modeling. The inter-arrival time corresponds to the arrival of video segments. The (unlimited) waiting room translates to the limited video buffer, which is equipped with the two thresholds p and q controlling the segment request behavior, and consequently the inter-arrival time. More specifically, the arrival process is paused if threshold q is exceeded, until the queue filling level falls below p again. The service time of a request corresponds to the playback of a video segment and hence equals the segment's duration. Please note that for the sake of simplicity, the illustration in Figure 3.4 neglects the quality adaptation behavior. That is, depending on additional buffer-related thresholds or depending on a segment's download duration, the next segment is requested in a specific quality level. While this can take effect on the segment inter-arrival times, as the different bitrates along the distinct qualities impact the download duration, the service time is not affected by the different qualities.

Table 3.1 summarizes the notations used throughout the description of the model. The number of video quality levels can be arbitrary and is denoted as $N \in \mathbb{N}$. The quality level

Table 3.1: Notations used for describing the analytical HAS model.

INPUT	FIXED	N	number of provided quality levels
		$\mathcal{Q}$	quality levels $\mathcal{Q} = \{1,\ldots,N\}$
		q	threshold after which buffering is paused
		p	threshold for resuming buffering after having stopped before
		qt_i	threshold (buffer or rate) for requesting quality $i \in \mathcal{Q}$, $qt_1 = 0$
		Q_i	set of states (buffer or rate) requesting quality $i \in \mathcal{Q}$
	RANDOM VAR. (RV)	$A_n^{(i)}$	inter-arrival time RV of segment n of quality $i \in \mathcal{Q}$
		B_n	playback time RV of segment n
		$C_n^{(i)}$	average bitrate RV of segment n of quality $i \in \mathcal{Q}$
		D_n	average throughput RV for downloading segment n
ANALYSIS	RANDOM VAR.	U_n	buffer RV *immediately after* arrival of segment $n-1$
		$\hat{U}_n$	virtual buffer RV *immediately before* arrival of segment n, which might attain negative values to denote stalling
		$\tilde{U}_n$	actual buffer RV ≥ 0 *immediately before* arrival of segment n

for the next segment is always decided immediately upon having received the previous video segment: based on the current buffer level or based on the latest download rate either of the quality levels is chosen. The decision is made according to pre-defined quality switching thresholds, denoted as qt_i. Given qt_i for $i \in \{2,\ldots,N\}$, the highest quality i is chosen such that the buffer level or download rate lies above qt_i ($qt_1 = 0$). Furthermore, the model allows for considering buffering thresholds p and q, $p < q$, where once the buffer level q is exceeded, the next video segment is delayed until the buffer undercuts the threshold p. This limits the overall amount of buffered video at any time and is typically done in HAS client implementations to avoid high bandwidth wastage in case the user aborts the playback.

Each quality level i corresponds to a (discrete) set $Q_i \subseteq \mathbb{N}$ of buffer levels or rates upon which quality level i is requested. Considering the buffer-based model, we define

$$\begin{aligned} Q_1 &= \{0,1,\ldots,qt_2-1\}, \\ Q_i &= \{qt_i,\ldots,qt_{i+1}-1\},\ i \in \{2,\ldots,N-1\}, \\ Q_N &= \{qt_N,\ldots,\max_{buf}\}, \end{aligned} \qquad \text{[buffer] (3.7)}$$

where $\max_{buf}$ denotes the maximal possible buffer size, e.g., the length of the video.

For the rate-based model, the sets are analogously defined, but using $\max_{rate}$ to denote the maximal rate:

$$\begin{aligned} Q_1 &= \{0,\ldots,qt_2-1\}, \\ Q_i &= \{qt_i,\ldots,qt_{i+1}-1\},\ i \in \{2,\ldots,N-1\}, \\ Q_N &= \{qt_N,\ldots,\max_{rate}\}. \end{aligned} \qquad \text{[rate] (3.8)}$$

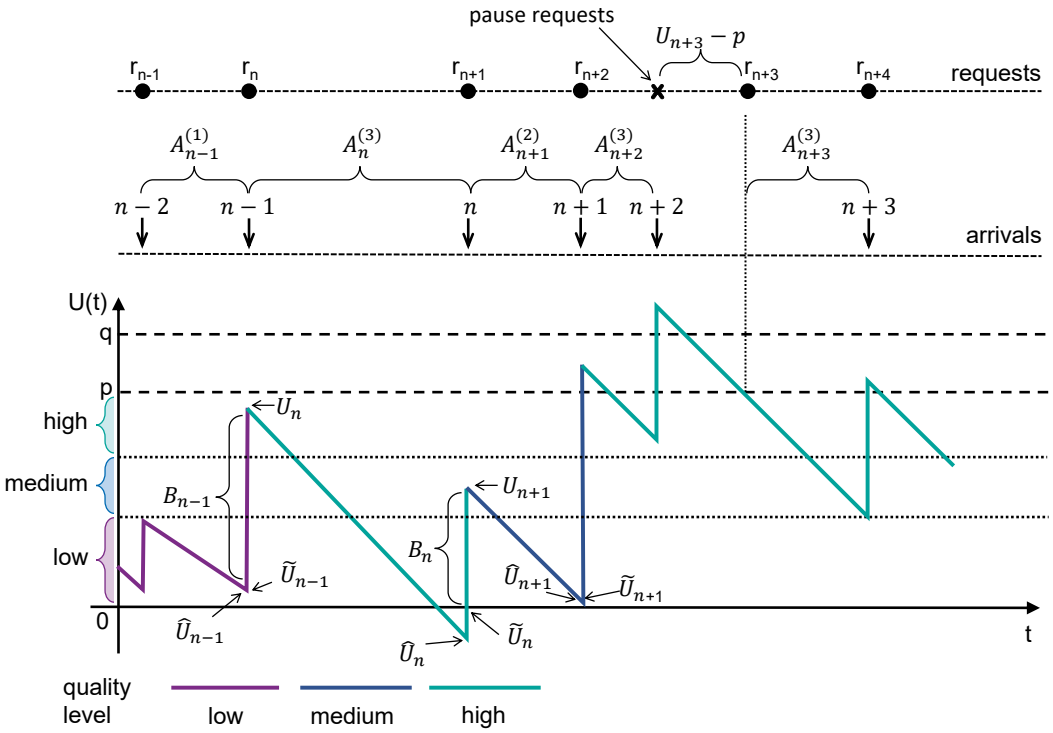


Figure 3.5: Sample state process of GI/GI/1 buffer with *pq*-policy and quality switching behavior. The quality is determined by a control variable, e.g., the estimated rate or the current buffer state, as in the shown in the illustration. Different line colors denote different quality levels.

While the above parameters are specific to the HAS implementation and assumed to be fixed, the external factors, as for example the available bandwidth, are modeled as *random variables* (RV) drawn from *arbitrary* probability distributions. Specifically, we denote by B_n the random variable determining the duration of segment n. As HAS services typically utilize the same duration for each video segment, we note that this behavior can easily be modeled by using a deterministic, fixed distribution for the segment play time. D_n denotes the average throughput while downloading segment n. The bitrate for segment n of quality $i \in \{1,\ldots,N\}$ is modeled as the RV $C_n^{(i)}$. Accordingly, the inter-arrival time, i.e., the download duration, of segment n of quality i, can be computed as follows:

$$A_n^{(i)} = \frac{C_n^{(i)} \cdot B_n}{D_n}. \tag{3.9}$$

Note that the distribution of $A_n^{(i)}$ has to be calculated by a ratio distribution in order to consider the bitrate of the segment and the download bandwidth. Since we are utilizing discrete-time analysis, we can compute the ratio distribution by iterating over all possible combinations.

Figure 3.5 gives an exemplary overview of the model using three quality levels, which means that $N = 3$ and $i = 1,2,3$. The time t is depicted along the x-axis and the y-axis

represents the buffered playback time. In the example, for the $n-1$-th segment, quality level $i=1$ is chosen based on the control set Q_1. Please note that the illustrative example denotes the buffer ranges for selecting a specific quality level, with low corresponding to $i=1$, medium corresponding to $i=2$, and high corresponding to $i=3$, respectively. The quality can, however, similarly be chosen depending on the download rate, without any loss of generality. After the full download time of $A_{n-1}^{(1)}$ (but before the segment arrival) two buffer levels are considered $\hat{U}_{n-1}$ and $\tilde{U}_{n-1}$: the virtual buffer value $\hat{U}_{n-1}$ allows for negative values, which will indicate stalling, while $\tilde{U}_{n-1}$ denotes the actual buffer value, which can at most be empty. Specifically, $\hat{U}_{n-1}$ is obtained by subtracting the download time $A_{n-1}^{(1)}$ from U_{n-1}, whereby U_{n-1} denotes the buffer state immediately after reception of the $n-2$-th segment. The actual buffer value $\tilde{U}_{n-1}$ is obtained by considering the maximum of $\hat{U}_{n-1}$ and 0. Hence, as $\hat{U}_{n-1} \geq 0$ holds in the example, $\tilde{U}_{n-1} = \hat{U}_{n-1}$ follows. Accordingly, the segment's playtime B_{n-1} is added to the buffer level $\tilde{U}_{n-1}$ to obtain the new buffer level U_n immediately after receiving the $n-1$-th segment. According to the control sets, the next segment is downloaded in the highest quality $i=3$, resulting in a download duration of $A_n^{(3)}$. As the download duration $A_n^{(3)}$ is larger than the current buffer time U_n, the virtual buffer $\hat{U}_n$ drops below zero while for the actual buffer $\tilde{U}_n = 0$ holds. In this case, the video playback is stalled for a duration of $-\hat{U}_n$ time units. After receiving the n-th segment, the actual buffered playback time is set to B_n, i.e., $U_{n+1} = B_n$ holds, and the playback resumes as the buffer value is now again positive. After downloading segment $n+1$ in quality $i=2$, the threshold q is exceeded. Thus, requesting segment $n+2$ is delayed until the buffer drops below the threshold p.

Finally, we note that the model relies on the following assumptions:

1. Round trip times and protocol overhead are not modeled.
2. Segments must be fully received before being played back.
3. After a video stalling, the playback is immediately resumed once a segment arrives: there is no buffer limit for resuming the playback.
4. Segment arrivals A_n and the service time B_n are required to be independent probability distributions.

3.2.2 Stochastic Preliminaries

The above presented model essentially represents a complex $\mathsf{GI}/\mathsf{GI}/1$ queuing model. More specifically, we can transfer the specifics of the $\mathsf{GI}/\mathsf{GI}/1$ model with bounded delay, which only accepts requests up to a certain level of remaining workload in the system, to the buffer- and quality switching policies of the presented model. While in the example of Figure 3.5 the *random* variables, e.g., $A_n^{(i)}$ and U_n attained specific values, the model will work on the *distributions* of these random variables. We denote the underlying probability density functions by employing lower case letters: $a_n^{(i)}$, b_n, $c_n^{(i)}$, and d_n denote the density functions of the input random variables $A_n^{(i)}$, B_n, $C_n^{(i)}$, and D_n. While these densities are provided as

part of input, the main task when analyzing the model is to concisely compute the density functions pertaining to the buffer level random variables U_n, $\tilde{U}_n$, etc., as these are used to compute the quality metrics.

In order to take the latest buffer state or download rate into account, we use the σ-operators σ^m from Equation 3.5 and σ_m from Equation 3.6 to truncate the respective probability distributions to a certain range.

To account for negative buffer values, we apply the π_m operator from Equation 3.4 with $m = 0$ to concentrate all probability mass below 0 to 0. Accordingly, to account for buffer values exceeding the threshold p, we introduce the sweep operator π^p.

$$\pi_0\big(f(k)\big) = \begin{cases} f(k), & k > 0 \\ f(0) + \sum_{j<0} f(j), & k = 0 \\ 0, & k < 0 \end{cases} \tag{3.10}$$

$$\pi^p\big(f(k)\big) = \begin{cases} f(k), & k < p \\ f(p) + \sum_{j\geq p} f(j), & k = p \\ 0, & k > p \end{cases} \tag{3.11}$$

3.2.3 Buffer-based Model

In the following the buffer-based model is presented. The core challenge in modeling the buffer-based behavior lies in computing the current buffer levels $\hat{U}_n$, $\tilde{U}_n$, and U_n. The computation of these is necessitated as the quality is chosen based on the buffer levels U_n and for example $\hat{U}_n$ will be used to compute the stalling probability. For our analysis we assume $qt_N \leq p < q$ to hold: if $qt_N > p$ was to hold, then the highest quality could never be requested after pausing buffering.

Henceforth, our main goal is to derive concise (recursive) formulas for the probability density functions u_n, $\hat{u}_n$, $\tilde{u}_n$ of the respective buffer level random variables. We start by considering the density function of $\hat{u}_n$ pertaining to the random variable $\hat{U}_n$ of the virtual buffer. $\hat{U}_n$ is generally computed as a function of the previous buffer level U_n just after receiving segment $n-1$ and the arrival time $A_n^{(i)}$ of the segment n of quality i. Hence, both the chosen quality i, dictating the download time $A_n^{(i)}$, and potentially pausing the buffering depend on the buffer level U_n and the following cases have to be considered:

$$\hat{U}_n = \begin{cases} U_n - A_n^{(i)} \text{ if } qt_i \leq U_n < qt_{i+1}, i \in \mathscr{Q} \setminus \{N\}, & (3.12) \\ U_n - A_n^{(N)} \text{if } qt_N \leq U_n < q\,, & (3.13) \\ p - A_n^{(N)} \text{ if } q \leq U_n\,. & (3.14) \end{cases}$$

Above, the first $N-1$ cases of Equation 3.12 capture the system's behavior when the buffer level U_n indicates requesting a quality contained in $\{1,\ldots,N-1\}$. When the highest quality

is chosen, the two cases of Equations 3.13 and 3.14 need to be considered: if the buffer level lies below q, then simply the next segment is downloaded according to the highest quality, while if q is exceeded, requesting the next segment is delayed until the buffer has reached the threshold p again (cf. Figure 3.5).

Considering the first $N-1$ cases, the density of $U_n - A_n^{(i)}$ computes to $\hat{u}_n(k) = u_n(k) * a_n^{(i)}(-k)$. However, as this only holds for the case $qt_i \leq U_n < qt_{i+1}$, only the following *partial* density functions are obtained:

$$\hat{u}_{n,i}(k) = \sigma_{qt_i}\Big(\sigma^{qt_{i+1}}\big(u_n(k)\big)\Big) * a_n^{(i)}(-k),\ i \in \mathcal{Q}\setminus\{N\}. \tag{3.15}$$

Above, the restriction on the *range* of U_n is realized by truncating the distribution $u_n(k)$ to the respective range between qt_i and qt_{i+1} using the respective σ-operators.

In a similar fashion, the density for the case of Equation 3.13 can be captured by the following partial density function:

$$\hat{u}_{n,N,<q}(k) = \sigma_{qt_N}\Big(\sigma^{q}\big(u_n(k)\big)\Big) * a_n^{(N)}(-k). \tag{3.16}$$

Lastly, for Equation 3.14 the following density is obtained:

$$\hat{u}_{n,N,\geq q}(k) = \pi^{p}\Big(\sigma_{q}\big(u_n(k)\big)\Big) * a_n^{(N)}(-k)\,. \tag{3.17}$$

Intuitively, by first applying the σ_q-operator only buffer levels exceeding q are considered, while the π^p operator then realigns the full probability mass to the value p.

Overall, the density function $\hat{u}_n$ computes to the *summation* over the *partial* densities of Equations 3.15 to 3.17 (cf. [96]). Hence, the following is obtained:

$$\hat{u}_n(k) = \begin{pmatrix} \sum_{i=1}^{N-1}\left(\sigma_{qt_i}\Big(\sigma^{qt_{i+1}}\big(u_n(k)\big)\Big) * a_n^{(i)}(-k)\right) \\ \quad + \quad \left(\sigma_{qt_N}\Big(\sigma^{q}\big(u_n(k)\big)\Big) * a_n^{(N)}(-k)\right) \\ \quad + \quad \left(\pi^{p}\Big(\sigma_{q}\big(u_n(k)\big) * a_n^{(N)}(-k)\Big)\right) \end{pmatrix}. \tag{3.18}$$

Given the ability to compute the density $\hat{u}_n$ we can easily derive the densities of $\tilde{U}_n$ and U_n. Considering $\tilde{U}_n$, we note that $\tilde{U}_n = \max\{0, \hat{U}_n\}$ holds, as the actual buffer can never attain a negative value. Hence, the following density function is obtained using the π_0 sweep operator:

$$\tilde{u}_n(k) = \pi_0\big(\hat{u}_n(k)\big)\,. \tag{3.19}$$

Lastly, as the buffer level U_{n+1} *immediately after* the arrival of segment n computes to $U_{n+1} = \tilde{U}_n + B_n$, the convolution of the respective densities is employed to obtain:

$$u_{n+1}(k) = \tilde{u}_n(k) * b_n(k)\,. \tag{3.20}$$

Given a specific density function u_n together with the input densities $a_n^{(i)}(k)$ and $b_n^{(}k)$, the density function u_{n+1} can be computed using the Equations 3.18 to 3.20. Hence, given an initial distribution for u_1 (modeling the empty buffer), the exact density functions for *any* segment can be readily computed.

Figure 3.6 summarizes the computation of the buffer distribution. We start with the density of the video buffer immediately after arrival of segment $n-1$, i.e., $u_n(k)$ ①. As the quality, and hence the download duration, depends on the specific buffer values, we need to consider the different cases, as indicated in Equations 3.12 to 3.14. More specifically, if the buffer exceeds quality switching threshold qt_1, which is always the case as we set $qt_1 = 0$, but does not exceed qt_2 ②, the lowest quality, i.e., $i = 1$ is chosen. Accordingly, the density of the inter-arrival time for segment n is $a_n^{(1)}(k)$, which is subtracted from $u_n(k)$ by performing the convolution with $a_n^{(1)}(-k)$.

This is done analogues for the values of $u_n(k)$ exceeding qt_2, but undercutting qt_3, where the second quality level ($i = 2$) is chosen and the convolution is done using $a_n^{(2)}(-k)$ to account for the respective download duration ③. Hence, as long as any quality level below the highest quality ($i = N$) is chosen, Equation 3.12 is applied and the partial densities as denoted in Equation 3.15 are obtained.

If the buffer suffices for downloading the highest quality ($i = N$), two further cases have to be distinguished. In the first case ④, threshold qt_N is reached, but the buffer is still below q, the threshold for pausing segment request. It is hence sufficient to perform the convolution to account for the download duration for the highest quality segment $a_n^{(N)}(-k)$. For instance, Equation 3.13 is applied, resulting in the partial density as computed according to Equation 3.16.

In the second case ⑤, threshold q is exceeded, resulting in pausing segment requests until the buffer falls below threshold p. To account for this, all probability mass above q is concentrated to p using the sweep operator π^p. According to Equation 3.14, the partial density as denoted in Equation 3.17 is obtained.

To derive the virtual buffer density ($\hat{u}_n(k)$), all partial density function are summed up (Equation 3.18) ⑥. When applying the sweep operator π_0 ⑦ to the virtual buffer ($\hat{u}_n(k)$), the actual buffer prior to the arrival of the n-th segment ($\tilde{u}_n(k)$), which does not allow for negative values, is obtained. Finally, the downloaded segment's play time is added to the buffer, which is obtained by performing the convolution of $\tilde{u}_n(k)$ with the segment duration density $b_n(k)$ ⑧ as denoted in Equation 3.20.

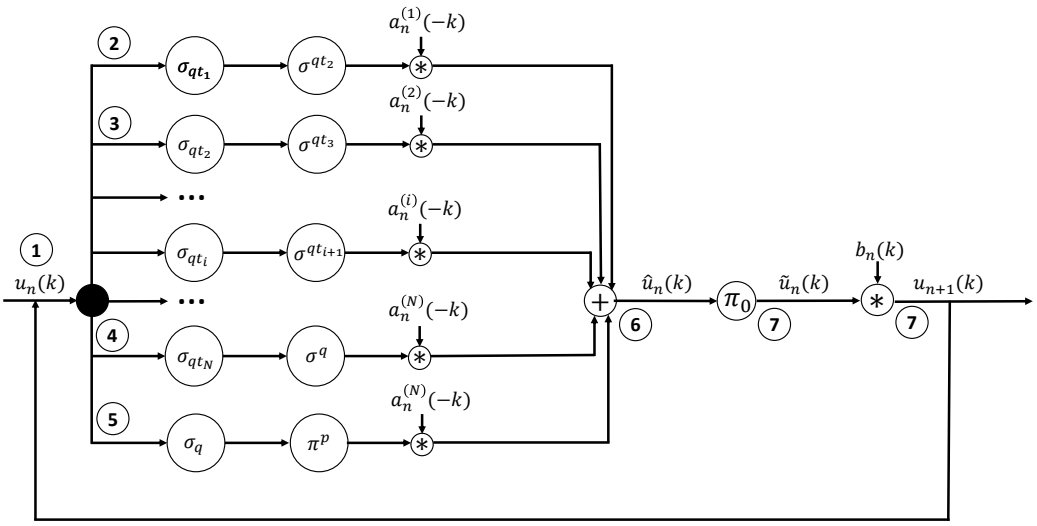


Figure 3.6: Computational diagram for the buffer-based version of the proposed model.

3.2.4 Rate-based Model

We now turn to the rate-based model. Again, the goal is to derive the probability density functions u_n, $\hat{u}_n$, $\tilde{u}_n$. The main difference with respect to the buffer-based model is that the quality selection now only depends on the last observed data rate D_{n-1}: if and only if $D_{n-1} \in Q_i$ holds, then the segment n is requested with quality i. Additionally, the buffer thresholds p and q are to be enforced: once U_n exceeds q, requesting further segments is delayed until the buffer reaches the lower threshold p again.

Given the fixed probability distribution d_n for the download rate of the n-th segment, the probability that a segment will be requested using a specific quality computes to

$$P(D_{n-1} \in Q_i) = \sum_{k \in Q_i} d_{n-1}(k) \quad \text{for} \quad i \in \mathcal{Q}.$$

Analogous to the analysis of the buffer-based model, we start by deriving the densities of $\hat{u}_n$ and $\tilde{u}_n$, which will then again be used to compute u_{n+1}. Regarding $\hat{u}_n$, fewer cases need to be considered (cf. Equations 3.12 to 3.14), as the requested quality only depends on the previous data rate. Nevertheless, a distinction has to be made depending on whether the buffer exceeds the threshold q. For $U_n < q$, the density computes to:

$$\hat{u}_n^{<q}(k) = \sum_{i \in \mathcal{Q}} P(D_{n-1} \in Q_i) \cdot \left(\sigma^q\big(u_n(k)\big) * a_n^{(i)}(-k) \right). \tag{3.21}$$

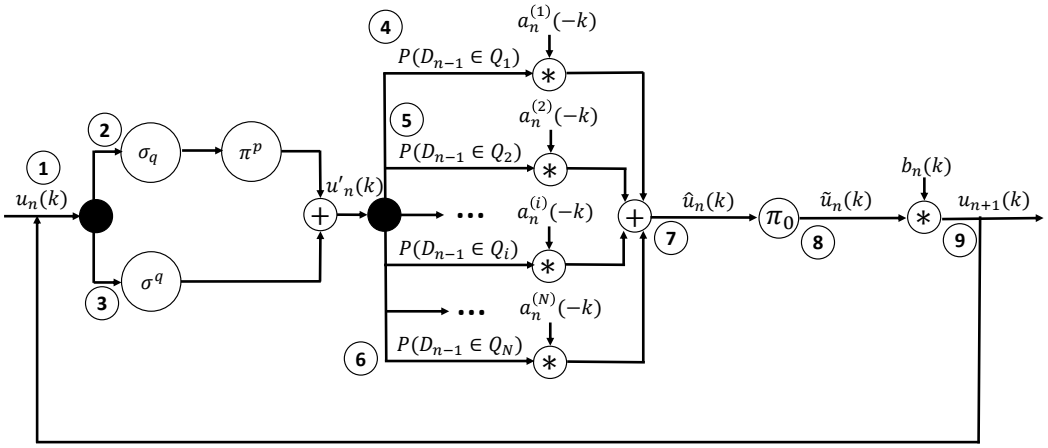


Figure 3.7: Computational diagram for the rate-based version of the proposed model.

In the other case, i.e., for $U_n \geq q$, the request is delayed until buffer level p is reached, and the density computes to:

$$\hat{u}_n^{\geq q}(k) = \sum_{i \in \mathcal{Q}} P(D_{n-1} \in Q_i) \cdot \left(\pi^p \left(\sigma_q \left(u_n(k) \right) \right) * a_n^{(i)}(-k) \right). \tag{3.22}$$

To merge the above Equations 3.21 and 3.22, note that both only differ in whether the π^p operator is applied or not and their conditioning. Accordingly, to unify both, we introduce the following buffer distribution u'_n, where the probability mass above q is realigned to p:

$$u'_n(k) = \sigma^q \left(u_n(k) \right) + \pi^p \left(\sigma_q \left(u_n(k) \right) \right). \tag{3.23}$$

Using this adapted density function u'_n, the density $\hat{u}_n$ of the virtual buffer $\hat{U}_n$ computes to:

$$\hat{u}_n(k) = \sum_{i \in \mathcal{Q}} P(D_{n-1} \in Q_i) \cdot \left(u'_n(k) * a_n^{(i)}(-k) \right). \tag{3.24}$$

Lastly, we note that the computation of the densities $\tilde{u}_n$ and u_{n+1} only rely on the computation of $\hat{u}_n$ (cf. Equations 3.19 and 3.20). Accordingly, the respective formulas still hold for the rate-based model, albeit now using $\hat{u}_n$ defined in Equation 3.24:

$$\tilde{u}_n(k) = \pi_0 \left(\hat{u}_n(k) \right), \tag{3.25}$$

$$u_{n+1}(k) = \tilde{u}_n(k) * b_n(k). \tag{3.26}$$

Figure 3.7 summarizes the above observations and depicts the iterative computation of the buffer densities. We start with the density of the video buffer immediately after arrival of segment $n-1$, i.e., $u_n(k)$ ①. The distinction needs to be made, whether threshold q is exceeded or not. If so ②, the sweep operator π^p is applied to shift all probability mass above

q to p. Otherwise, the buffer is below q ③ and no further operations need to be performed. Both partial densities for the buffer are then summed up and according to Equation 3.23, density $u'_n(k)$ is obtained, where the buffer cannot exceed p.

Next, the quality and hence the download duration for segment n is determined according to the rate observed during download of segment $n-1$, i.e., D_{n-1}. The probability for requesting the lowest quality level ($i=1$) is expressed as $P(D_{n-1} \in Q_1)$ ④ and is equivalent to the probability mass of d_{n-1} being below qt_2. In this case, the convolution of $u'_n(k)$ is performed with the density of the download duration to decrease the buffer by $a_n^{(i)}(k)$. Analogous to, the second quality level ($i=2$) is chosen for throughput values above qt_2 but below qt_3, corresponding $P(D_{n-1} \in Q_2)$ ⑤, as well as for any quality level i up to $i=N$ ⑥.

All partial densities are the summed up ⑦, as denoted in Equation 3.24, resulting in the virtual buffer density $\hat{u}_n(k)$. The remaining steps are equivalent to the buffer-based version of the model: The sweep operator π_0 shifts all probability mass below 0 to 0 so to obtain the buffer density prior to the arrival of segment n, i.e., $\tilde{u}_n(k)$ ⑧. Performing the convolution with the play time of the n-th segment, i.e., $b_n(k)$, finally results in the density $u_{n+1}(k)$, describing the buffer immediately after reception of the segment ⑨.

3.2.5 Metric Computation

In the following, relevant HAS performance metrics are formulated based on the above derived density distributions. While the model generally allows for studying the performance before and after the n-thsegment arrival (by iteratively computing the respective densities), in the following only steady state performance metrics will be considered. To this end, we have to assume that the input probability distributions are either constant, i.e., that $b_n = b$, $c_n^{(i)} = c^{(i)}$, and $d_n = d$ hold for $n \geq 0$, or that the density series $\{b_n\}_n$, $\{c_n^{(i)}\}_n$, and $\{d_n\}_n$ converge, such that the steady state density functions $b = \lim_{n\to\infty} b_n$, etc., are well-defined. In either of these cases, the buffer density functions will eventually converge as well. In the following, we assume $\lim_{n\to\infty} u_n = u$ and first consider the performance metrics related to the buffer state and then discuss metrics pertaining to the quality.

3.2.5.1 Buffer-related Metrics

Since a precise computation of the buffer level at any point in time is cumbersome and involves the computation of recurrence times [109], we rely on the buffer level u immediately after the segment arrival times, for computing the average buffer filling level. This metric can easily be computed, both for model and measurements. Accordingly, we estimate the average buffer level (at the segment arrivals) as:

$$\text{AVERAGE BUFFER LEVEL:} \qquad E(u) = \sum_k k \cdot u(k)\,. \tag{3.27}$$

Next, we consider the stalling probability and stalling duration. As discussed above, whenever $\hat{U}_n$ lies below 0, stalling of exactly $-\hat{U}_n$ time units occurs. As stalling only depends on the buffer distribution, we do not need to distinguish between the buffer- and rate-based approach. Considering the steady state probabilities, the stalling probability computes to:

$$\text{STALLING PROBABILITY:} \qquad P_{stalling} = \sum_{k<0} \hat{u}(k)\,. \tag{3.28}$$

To compute the stalling duration, we consider the negated expected value of $\hat{u}$ conditioned on the negative buffer levels:

$$\text{STALLING DURATION:} \qquad L = -\sum_{k<0} k \cdot \hat{u}(k)\,. \tag{3.29}$$

3.2.5.2 Quality-related Metrics

In contrast to the above presented metrics, to model the quality-related metrics, we have to distinguish between the buffer- and rate-based version for computing the quality-related metrics. They depend on the buffer levels in the buffer-based case and on the download durations in the rate-based case.

We first consider the average quality, which we define to lie in the range from 1 to N according to the definition of quality levels $\mathcal{Q}$. In the buffer-based model, the quality is dictated by the buffer level after having received a segment, i.e., u. For the rate-based model, the average quality is only determined by the steady state download rate d. Accordingly, we obtain:

AVERAGE QUALITY:

$$\bar{Q}_{buffer} = \sum_{i \in \mathcal{Q}} \left(i \cdot \sum_{k \in Q_i} u(k) \right), \tag{3.30}$$

$$\bar{Q}_{rate} = \sum_{i \in \mathcal{Q}} \left(i \cdot \sum_{k \in Q_i} d(k) \right). \tag{3.31}$$

Another performance metric of interest is the stability of quality levels in terms of the relative number of segments witnessing a quality level change. In the buffer-based case, the quality switches between the n-th and the $n+1$-th segment, if the buffer at the arrival of the n-th segments exceeds a threshold q_t, while at the arrival of the n-th segment, any other threshold than q_t is reached. The same holds for the rate-based approach, however, the threshold q_t refers to the data rate during download of the n-th and $n+1$-the segment, respectively. This can be translated for the (steady state) probabilities of switching quality levels as follows:

SWITCHING PROBABILITY:

$$P^{switch}_{buffer} = \lim_{n\to\infty} \sum_{i\in\mathcal{Q}} P(U_n \in Q_i, U_{n+1} \notin Q_i)\,, \tag{3.32}$$

$$P^{switch}_{rate} = \lim_{n\to\infty} \sum_{i\in\mathcal{Q}} P(D_n \in Q_i, D_{n+1} \notin Q_i)\,. \tag{3.33}$$

We first derive a concise formula for the buffer-based model, i.e., P^{switch}_{buffer}. As analyzed in Section 3.2.3, the random variable U_{n+1}, and respectively its density, depends on U_n (cf. Equation 3.20). As the different 'computational branches' of the computation of u_{n+1} (cf. Figure 3.6) need to be taken into account, we first decompose the switching probability P^{switch}_{buffer} into the sum:

$$\lim_{n\to\infty} \begin{pmatrix} \sum_{i=1}^{N} P(U_n \in Q_i,\ U_{n+1} \notin Q_i,\ U_n < q) \\ +\ \ P(U_n \in Q_N, U_{n+1} \notin Q_N, U_n \geq q) \end{pmatrix}. \tag{3.34}$$

Now, conditioning on the quality level of U_n and considering the steady state distribution u, the probability of quality switches for the buffer-based case P^{switch}_{buffer} can be computed as follows:

$$P^{switch}_{buffer} = \begin{pmatrix} \sum_{i=1}^{N} \sum_{k\notin Q_i} \left(\pi_0[\sigma_{qt_i}(\sigma^{qt_{i+1}}(u(k))) * a^{(i)}(\text{-}k)] * b(k)\right) \\ +\ \sum_{k\notin Q_N} \left(\pi_0[\pi^p(\sigma_q(u(k))) * a^{(N)}(\text{-}k)] * b(k)\right) \end{pmatrix}. \tag{3.35}$$

Intuitively, above the σ-operators represent the conditioning of U_n having a specific quality (of steady state density u), while the inner formulas are obtained by following the computational specification to obtain the buffer level probabilities for U_{n+1} and then considering all buffer levels k where another quality level was chosen ($k \notin Q_i$).

For the rate-based approach, the switching probability only depends on the (steady state) download rate. Accordingly, it can be readily computed using basic probability theory as follows:

$$P^{switch}_{rate} = \lim_{n\to\infty} \left(\sum_{i=1}^{N} P(D_n \in Q_i, D_{n+1} \notin Q_i)\right) \tag{3.36}$$

$$= \lim_{n\to\infty} \left(1 - \sum_{i=1}^{N} P(D_n \in Q_i) \cdot P(D_{n+1} \in Q_i)\right) \tag{3.37}$$

$$= 1 - \sum_{i\in\mathcal{Q}} \Big(\sum_{k\in Q_i} d(k)\Big)^2. \tag{3.38}$$

Table 3.2: Model input and measurement parameters used throughout the validation study.

Parameter	Value
maximum buffer	40 seconds
buffer-based thresholds	$qt_1 = 0s$, $qt_2 = 10s$, $qt_3 = 20s$, $qt_4 = 30s$
rate-based thresholds	$qt_i = 1.15$-fold of average bitrate of level i
bandwidth provisioning factor	$a = \{0.8, 1.0, 1.2, 1.4, 1.6, 2.0\}$-fold of average bitrate of the lowest quality level ($i = 1$)
bandwidth coefficient of variation	$c_v = 0, 0.2, 0.4, 0.6$
average video bitrates	br_i = {563, 1098, 1634, 2170} kbps for $i = 1, 2, 3, 4$

3.3 Model Validation

In the following, we validate the probabilistic outputs of the model. Therefore, we run testbed measurements and apply both versions of the model with the same properties used during the measurements, i.e., number and characteristics of quality levels, switching thresholds, etc. We first present the applied experiment paramters and discuss the results afterwards.

3.3.1 Methodology and Experiment Parameters

We generate video sequences using a virtual testbed setup. It consists of a video server and a HAS client running the TAPAS player [110]. The server and the client run in two different Linux namespaces[1] on a single physical machine. A virtual interface connects both namespaces and allows to throttle the bandwidth between client and server using the Linux traffic control[2]. As a rate-based HAS implantation, we use PANDA [53]. For the buffer-based approach, we use an own and very simple heuristic, which selects the quality based on the current buffer state and pre-defined thresholds.

The test video sequence consists of 48 segments with a duration of 5 seconds each, which are provided in four different quality levels, as denoted in Table 3.2. For the buffer-based approach, we set the quality switching thresholds to $qt_1 = 0$, $qt_2 = 10$, $qt_3 = 20$, and $qt_4 = 30$. In case of the rate-based approach, we set the safety margin, i.e., the percentage to which the throughput rate has to exceed the bitrate of a certain quality level, to 15%. Accordingly, the obtained thresholds are computed using the average bitrate per quality level as $qt_i = 1.15 \cdot C^{(i)}$, for $i \in \{2, 3, 4\}$, with $C^{(i)}$ denoting the average bitrate of quality level i. For the first threshold $qt_1 = 0$ holds. The maximum buffer threshold is set to $q = 40$. As we do not consider a segment request resume threshold, we set $p = q = 40$. For the available average bandwidth, we define the bandwidth provisioning factor a. It corresponds to the ratio of the average bandwidth to the average bitrate of the lowest quality level. A fluent video playback is generally only possible if $a > 1$. Additionally, we consider different settings of the

[1] http://man7.org/linux/man-pages/man7/namespaces.7.html
[2] https://linux.die.net/man/8/tc

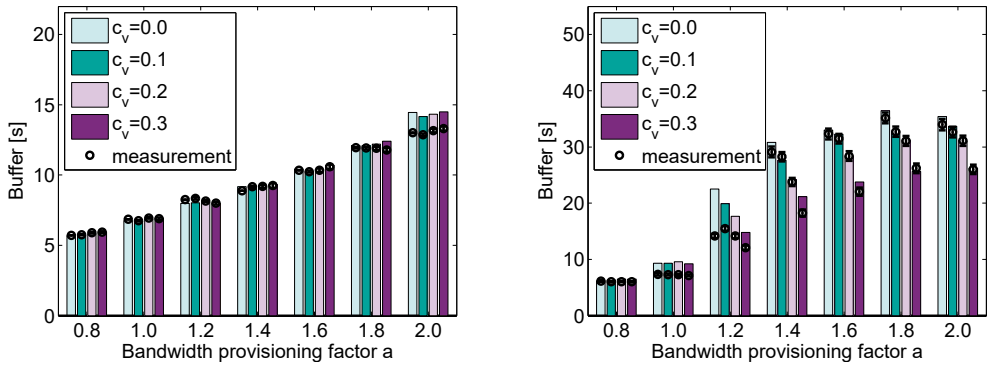
coefficient of variation c_v for the available bandwidth. To enable measurements with specific parameter settings for a and c_v, custom network trace files are generated. Thereby, the values for the available bandwidth per second are chosen according to a negative binomial distribution, so to achieve the target bandwidth characteristics.

Our analytical computations are based on the same empirical distributions and values as used for the measurements. With respect to the video characteristics, we compute the average values of the bitrates and the standard deviations and approximate the empirical distribution with a negative binomial distribution. This allows us to abstract the specific video clip and to generalize our results. For the rate distribution, needed as an input for the analytical model, we utilize the empirical distribution from the testbed client's rate estimation. The client estimates the rate by dividing each received segment's size by its download duration. Typically, the download takes roughly as long as a segment's duration. Hence, the actual throughput for a segment is the average of several random bandwidth values specified in the trace files. As a result, the coefficients of variation for the estimated rate, which are considered in the model, are lower than the coefficients of variations for the available bandwidth, which are considered in the network traces for the measurements. For instance, c_v values of 0.2, 0.4, and 0.6 for the available bandwidth reduce to c_v values of roughly 0.1, 0.2, and 0.3 for the estimated rate. As the model considers the available bandwidth on a per-segment scale, we use the latter ones, i.e., the empirical client-side measurements as the input for the model. Again, by using a distribution to describe the network characteristics, temporal effects are eliminated.

For our evaluations, we only consider the buffer state immediately after segment arrival in both cases, when performing the testbed measurements and when applying the analytical model. Hence, the actual average buffers are expected to be slightly lower. In order to account for overhead such as MPD file or TCP/IP packet headers, we reduce the available bandwidth in the model by a factor of 0.07, matching the average overhead ratio measured in the testbed. Finally, we note that for each or the tested parameter combination, 15 testbed measurement runs are performed in order to obtain statistical significance.

3.3.2 Validation Results

First of all, we evaluate how accurate the average buffer filling level can be captured by the analytical model. Figure 3.8a shows the results for the buffer-based approach. The x-axis depicts the bandwidth provisioning factor, the y-axis denotes the buffer level in seconds. Values obtained from the analytical model are depicted as bars, whereas the colors express different c_v values. The dots represent the buffer levels obtained from the testbed measurements along with the 95% confidence intervals. It can be seen that the model is quite accurate for all evaluated c_v parameters and provisioning factors a. The highest deviations are observed for $a = 2$, where the model returns higher values for the average buffer for any of the examined c_v.



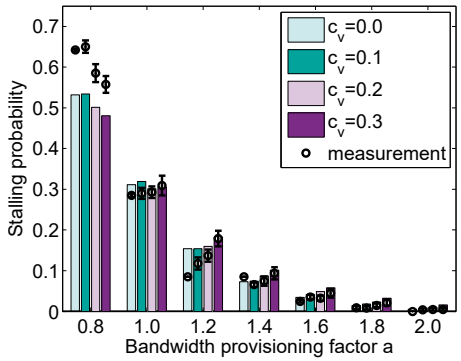
(a) Buffer-based model. **(b)** Rate-based model.

Figure 3.8: Comparison of analytical (bars) and empirical (dots) buffer levels with the 95% confidence intervals for the measurements.

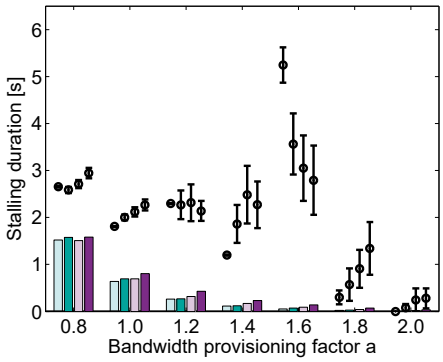
Analogous, Figure 3.8b depicts the buffer levels obtained for the rate-based model. For bandwidth provisioning factors of $a = 1.0$ and $a = 1.2$, we see discrepancies between the model's output and the testbed measurement. However, for the remaining bandwidth provisioning factors, the model can accurately capture the buffer level any considered values of c_v. For the rate-based version, it can also be observed that for $a \geq 2$, the average buffer decreases as c_v increases. This is due to the fact that higher segment qualities can be downloaded more often, resulting in a higher download duration, and hence decreasing the average buffer level.

Next, we study the QoE-IFs, which can be derived from the modeled video buffer. The results for the buffer-based version are depicted in Figure 3.9. As shown in Figure 3.9a, the analytical model in general yields a good approximation of the stalling probability obtained in the testbed measurements. Significant deviations can only be observed for two settings of the bandwidth provisioning factor a. For very low bandwidth settings, i.e., $a = 0.8$, the model under-estimates the stalling probability by an absolute value of about 0.1 for any c_v. The second setting where greater deviations occur is $a = 1.2$ with low values for c_v. In this case, the model over-estimates the stalling probability, but the deviation is in any case less than absolute value of 0.1.

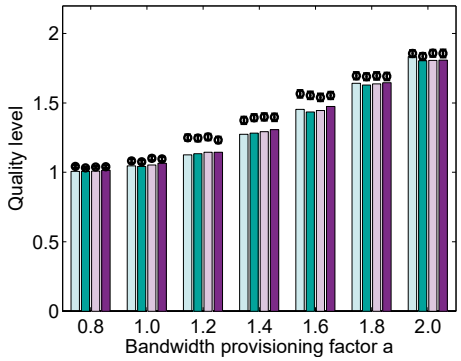
While the stalling probability can be modeled with high accuracy in most cases, there are larger discrepancies when it comes to the duration of stallings, as shown in Figure 3.9b. For each of the tested parameter settings, the analytical model under-estimates the stalling duration. For bandwidth provisioning factors of $a = 0.8$ and $a = 1.0$, the deviation is about 1.0s to 1.5s and reaches up about 5s for $a = 1.6$ and $c_v = 0$. We note here that the overall number of stallings occurring during one testbed video session is very low. In a single measurement run, only 48 segments are downloaded and if $a > 1.0$, stallings are rare events, leading to the small sample size. Hence, the average stalling duration is obtained from a low number of samples, which also explains the large confidence intervals. The issue can be overcome by performing more measurements to obtain a greater sample size and by introducing more randomness, e.g., by using different video sequences.



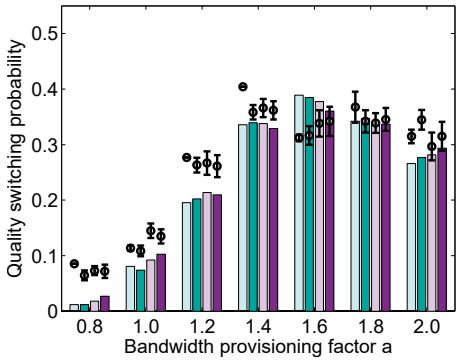
(a) Stalling probability.



(b) Avgerage stalling duration.



(c) Avgerage video quality level.



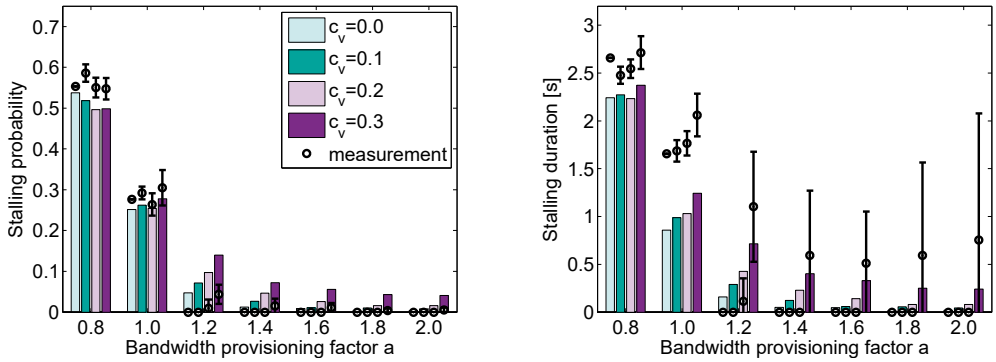
(d) Switching probability.

Figure 3.9: Buffer-based model: Comparison of analytical (bars) and empirical (dots) quality metrics with the 95% confidence intervals for the measurements.

Figures 3.9c and 3.9d depict the metrics related to the video quality, i.e., the average quality level and the quality switching probability. In general, the model slightly under-estimates the average quality, but is still capable to yield accurate results. In terms of the switching probability, we can observe certain deviations, but the model outputs still preserve the trends and the absolute differences between model and measurements are in case below 0.1.

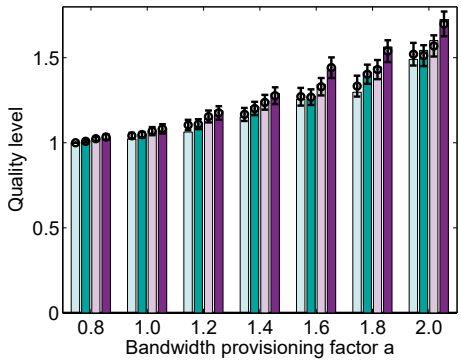
Accordingly, Figure 3.10 illustrates the HAS performance metrics for the rate-based version of the model. The stalling probability, denoted in Figure 3.10a, is relatively accurate modeled for $a = 0.8$ and $a = 1.0$, however, the model over-estimates the probability for video interruptions in the remaining cases. The maximum deviation occurs for $a = 1.2$ and $c_v = 0.3$, with an absolute drift of approximately 0.1.

Figure 3.10b denotes the stalling duration. We obtain similar outputs from measurements and model, with a deviation not exceeding 0.5s, for $a = 0.8$, i.e., the scenarios where stallings are likely to occur with high frequency. For $a = 1.0$, the model can preserve the trend of increasing stalling duration with increasing c_v, but the discrepancy between model and measurement is enlarged. Similar as with the buffer-based approach, due to the few stallings during the measurements when $a > 1.0$, i.e., the small sample size from which the average stalling duration is composed, the results from measurements and model deviate.

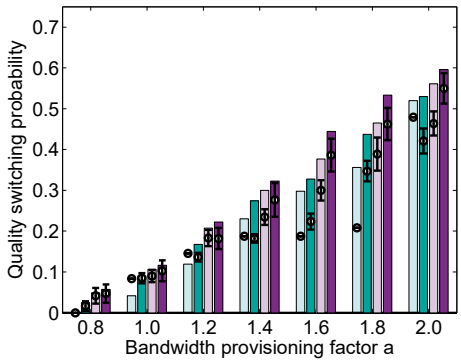


(a) Stalling probability.

(b) Avgerage stalling duration.



(c) Avgerage quality level.



(d) Switching probability.

Figure 3.10: Rate-based model: Comparison of analytical (bars) and empirical (dots) quality metrics with the 95% confidence intervals for the measurements.

Finally, we discuss the metrics related to the video quality. As shown in Figure 3.10c, the model yields highly accurate results for the average quality. For the quality switching probability denoted in Figure 3.10d, we observe that the model over-estimates how often quality switches occur, especially for $a > 1.2$.

The comparisons between model and measurements indicate that the model enables a good prediction of all metrics apart from the stalling durations. It should be noted that the measurements were conducted using a relatively short video, which in turn leads to a small sample size when evaluating metrics like the stalling duration, especially for higher values of a, where interruptions are per se unlikely to happen. A higher statistical significance in the measured results can be obtained by using a diverse set of video clips, longer video sequences, and by increasing the number of measurement runs. However, as the stallings occur only for small number of parameter configurations, obtaining a higher significance in terms of stalling duration is costly. Regarding the discrepancy between measurements and model, it should be noted that so far, we only apply one fixed share of 0.07 to account for overheads. However, this value is only a rough estimate and depends on the bandwidths and bitrate. Despite these and other assumptions, e.g., no influence of RTT or the independence of RVs, the results still indicate the applicability of the presented model.

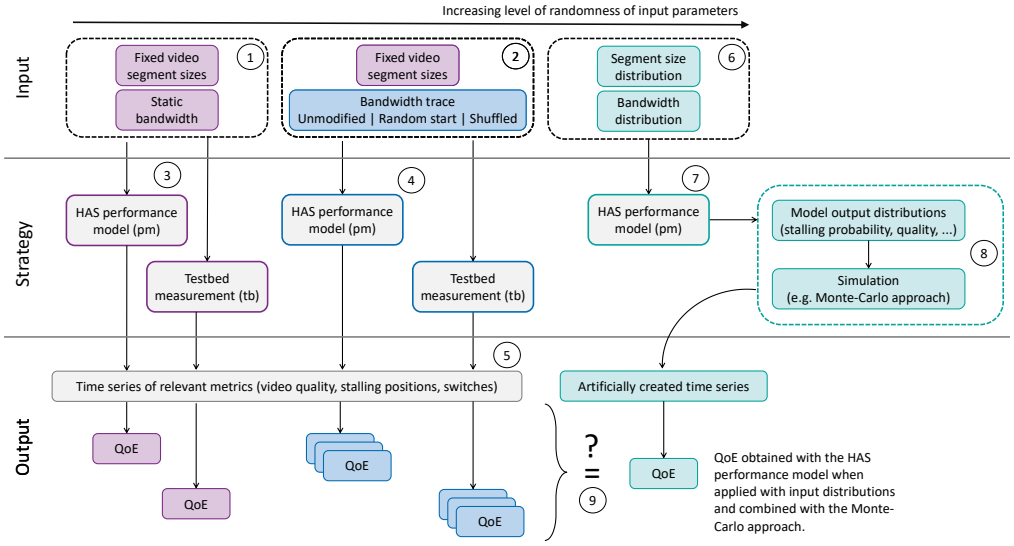


Figure 3.11: Overview of the applied methodology to assess the model's capability for obtaining reliable QoE output, despite its lack of temporal context.

3.4 Model Applicability for QoE Analysis

In this section, we examine in how far the model can be applied for QoE analysis. As outlined in the previous parts of this chapter, the analytical model returns asymptotic probabilities, but is not capable of capturing the temporal behavior of video sessions. However, state-of-the art QoE models require temporal aspects, such as the timely position of stallings. In the following we propose an approach based on Monte-Carlo simulation to generate video sequences from the model's output. These video sequences are then used to obtain the QoE via the P.1203 model and, similarly as the validation carried out above, they are compared against the QoE obtained from video sessions generated via testbed measurements. We first give a brief overview of the proposed methodology for video sequence generation and introduce the experimental parameters. The section concludes with a short discussion of the results.

3.4.1 Methodology Overview

A comprehensive overview of the applied methodology is illustrated in Figure 3.11. It shows the whole process with respect to the testbed measurement runs, model-based evaluations, obtaining time series of the video playback, and the QoE assessment using ITU-T P.1203. The top of the illustration shows the input in terms of segment sizes and bandwidth. We consider static bandwidths ① as well as bandwidths with varying capacity over time ②, by applying a real network trace [111], which denotes the bandwidth as a time-series on a time scale of one second. The bandwidth trace is used in three different manners. In the first case, the trace is not modified and starts at the beginning (*unmodified*). In the second case, we pick a random start point for the trace (*random start*). In the third case, we shuffle the

whole trace file (*shuffled*). Especially the third case, where the per-second bandwidth values are shuffled, increases the randomness by eliminating the temporal correlation in the trace file. The video segment sizes are fixed, i.e., there is a pre-defined, chronological order of the segments' sizes, Having these segment sizes, testbed measurements are performed and the buffer-based HAS performance model is applied, once assuming the static bandwidth ③ and once using bandwidth traces ④. As both, the segment sizes and the bandwidth values are fixed and in chronological order, the HAS performance model can be used to retrieve a time series of the video buffer. More specifically, the model is capable to compute the buffer state immediately before and after segment arrival. Hence, in both cases, when using the HAS performance model and when running the testbed measurements, we obtain a time series of all relevant HAS performance parameters ⑤, such as the playback quality or the number and duration of video interruptions.

In order to exploit the benefits of the analytical model, i.e., the efficient computation of relevant metrics for a huge number of parameter combinations, it has to be with input distributions, instead of chronological sequences. Hence, in a next step we increase the level of randomness of the input data and express the segment sizes and bandwidth as distributions ⑥, which follow the same statistical properties, i.e., mean values and standard deviations, as their respective chronological sequences. Thus, the distributions used in ⑥ are an abstraction of the time series used in ②. When the HAS performance model is applied with these distributions, it returns steady state results of relevant adaptive streaming metrics like stalling probability or the video buffer distribution ⑦. As we need video play back sequences in order to compute the QoE with the P.1203 model, we perform Monte-Carlo [112] simulations ⑧ to generate these sequences, as further described in Section 3.4.2.

Hence, we constantly increase the level of randomness of the input data, from fixed traces and time series to probability distributions and want to study the impact of disorder and abstraction on the obtained QoE scores. The time series obtained via measurement, via the model with trace inputs, and via using the model with distribution inputs in conjunction with Monte-Carlo simulations, are evaluated and compared with regard to the final QoE score, which is obtained by using the ITU-T P.1203 model. The ultimate goal is to evaluate in how far the HAS performance model, when it is applied in its most efficient manner, i.e., relying on distributions for segment sizes and bandwidth, can reliably capture the QoE. For that, we compare the obtained QoE scores against those scores obtained with the performance model with fixed inputs and against those obtained via testbed measurements ⑨.

3.4.2 Video Playback Sequence Generation

In this subsection we describe how we generate video time sequences based on the output of the HAS performance model, when applied with generic distributions for the segment sizes and available bandwidth (see ⑦ and ⑧ in Figure 3.11). To generate a couple of different video playback sequences, we use a Monte-Carlo simulation approach, namely a random

walk. We start with the empty system and compute the transmission time for the first segment. This is done by drawing a random inter-arrival time from the corresponding segment inter-arrival time distribution for the selected quality defined by the current buffer level and the quality switching thresholds. The video playback buffer depletes until the next segment arrives at the video client, which results in an increase of the video playback buffer by a random variable following the segment length distribution. Relevant QoE metrics, such as the downloaded video quality or the stalling time and duration, are logged. Then, the random walk is continued by drawing further random variables for inter-arrival times depending on the developing video buffer state, as well as random variables for the segment length distributions. This is continued until the desired number of segment downloads is reached. In total, we randomly generate 50 video playback sequences for each input parameter combination using the outlined method.

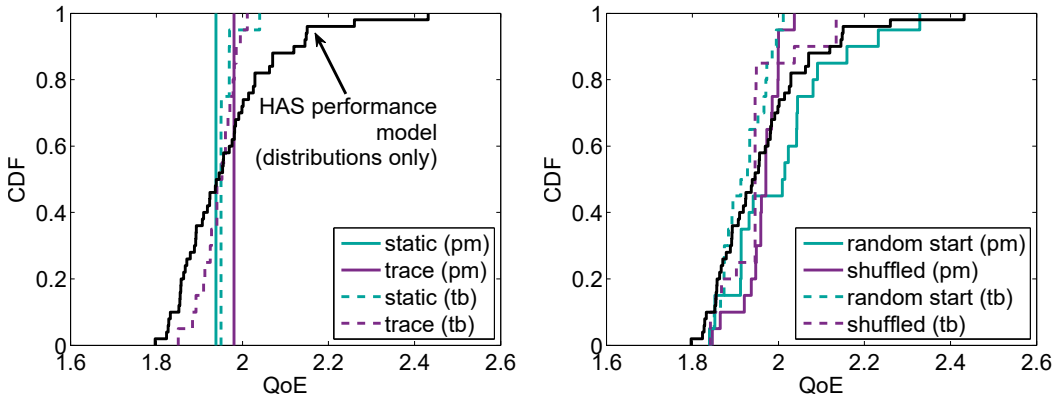
3.4.3 Experimental Parameters

Our test video with a total duration of 240 seconds is split into 60 video segments, each of 4 seconds duration. Each video segment is available in four different quality levels, with the following tuples (average bitrate, standard deviation) denoted in kbps: (510,154), (1016,313), (1524,474), (2034,634). For the available bandwidth, we consider on the one hand *static* limits of 500, 1000, and 2000 kbps, which do not change throughout the experiments. On the other hand, we use the *car* bandwidth trace [111], which we scale so to achieve similar average values as for the static case. The scaled trace files result in the following tuples for average bitrate and standard deviation denoted in kbps: (496,260), (992, 520), (1984, 1040). In any of the three cases, the coefficient of variation (c_{var}) for the available bandwidth is 0.52.

Please note that the selected bandwidth configurations are lower than what can be expected from a real (mobile) network deployment. However, the goal of the study is to investigate if the proposed methodology, i.e., generating video sequences via Monte-Carlo simulations from probability distributions, is capable to retrieve QoE scores comparable to those obtained from testbed measurements. Setting a comparably low bandwidth capacity ensures that video stallings occur, which are one of the most dominant factors impacting QoE.

The HAS performance model is once applied on time series and once with input distributions for the available bandwidth. In the latter case, we use a deterministic distribution for *static* bandwidth limits. For varying bandwidth, realized in the testbed measurements with the *trace* experiments, we use distributions corresponding the characteristics mentioned above, i.e., we preserve the same average values and standard deviations of the traces' bandwidth values.

For the testbed measurements, we use the buffer-based bitrate adaptation strategy Bola, which is implemented in the Dash.js player. The quality switching thresholds are configured as follows: With a buffer value below 10 seconds, the lowest quality level, i.e., level 1 is requested. A buffer value of at least 10 seconds is necessary for requesting level 2, and



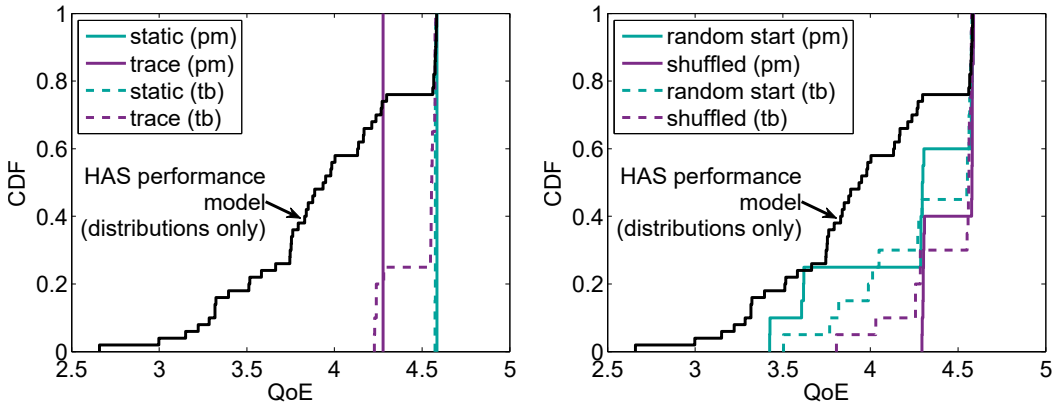
(a) Low level of randomness. **(b)** Increased level of randomness.

Figure 3.12: CDF of the QoE values resulting from the different bandwidth modes with an average bandwidth of 500 kbps. Black solid lines indicate the HAS performance model's outcome on distributions. The remaining solid lines indicate QoE values computed by the HAS performance model (pm) relying on time series as input. Dashed lines represent the results from measurements in the testbed (tb).

15 seconds of buffered play time triggers the heuristic to download the next segment in the 3rd quality level. Level 4 is requested if the video buffer exceeds a threshold of 20 seconds. When applying the HAS performance model, the quality switching thresholds are set accordingly.

3.4.4 Evaluation Results

We first evaluate the obtained QoE values for different bandwidth modes and levels of randomness with an average available bandwidth of 500 kbps. The results are depicted in Figure 3.12. More specifically, Figure 3.12a shows the results for the two bandwidth modes without or low level of randomness, i.e., the *static* bandwidth and the bandwidth *trace* started from the beginning. In case of *static*, the HAS performance model (pm) returns a MOS value of 1.94 and a MOS value of 1.98 when the bandwidth trace is applied. As the model computes stallings and the next segment's quality based on fixed inputs, the output is deterministic and we obtain only one MOS value for *trace* and one MOS value for *static*. The testbed measurements (tb), however, return slightly varying QoE values. In the case of *static*, the values range between 1.95 and 2.04. If testbed measurements are performed using *trace*, the variability increases slightly, returning MOS values between 1.85 and 2.0. These variations in the measurements are due to several aspects, including the behavior of TCP and software. When we apply the HAS performance model with distributions for bandwidth and segment sizes, it yields distributions for QoE-relevant metrics, such as stalling probability and video quality. Using the Monte-Carlo simulation technique (see Figure 3.11 ⑧), we generate 50 video sequences from these output distributions, i.e., we create chronological orders of video quality and staling events. The resulting MOS values range from 1.8 to 2.4, as shown by the black solid line. Hence, the MOS range, as obtained with the proposed



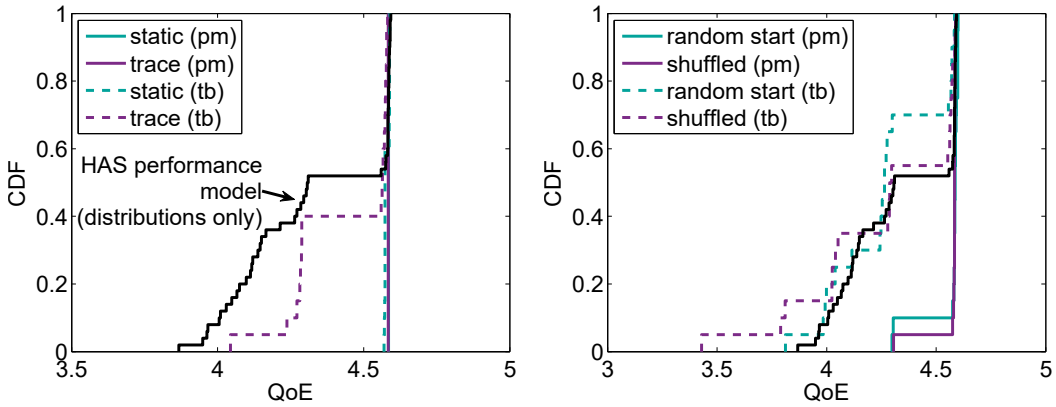
(a) Low level of randomness. **(b)** Increased level of randomness.

Figure 3.13: CDF of the QoE values resulting from the different bandwidth modes with an average bandwidth of 1000 kbps. Black solid lines indicate the HAS performance model's outcome on distributions. The remaining solid lines indicate QoE values computed by the HAS performance model relying on time series as input. Dashed lines represent the results from measurements in the testbed.

methodology covers all MOS scores obtained via testbed measurements. Moreover, it is notable that the median score obtained with the Monte-Carlo approach is in line with the median score obtained via testbed measurements.

The results when increasing the level of randomness by using random start points in the bandwidth trace or by shuffling the bandwidth trace are shown on in Figure 3.12b. While the computation using the performance model (pm) with time series for the bandwidth, i.e., *static* and *trace*, resulted in a single MOS value, we now obtain different QoE values for different computation runs, due to the induced randomness. The retrieved MOS lies between 1.84 and 2.3 for *random start* and between 1.85 and 2.0 for *shuffled*. The QoE as measured in the testbed (tb)ranges from 1.84 to 2.0 for *random start* and 1.84 and 2.1 for *shuffled*. Again, applying the model with input distributions and using the Monte-Carlo approach to generate time series for the video session, as indicated by the black solid line, is capable of fully covering the QoE ranges obtained from testbed measurements and using the model with input time series. Furthermore, the median obtained with the proposed methodology is similar to the median QoE scores from testbed measurements, indicating a good approximation.

The cumulative distribution functions for the modes with an average available bandwidth of 1000 kbps are shown in Figure 3.13. First, we discuss the results obtained when the bandwidth remains *static* or when using *trace* without any randomization, as shown in Figure 3.13a. Once again, we obtain a single QoE value when applying the HAS performance model (pm) with fixed time series input, giving MOS scores of 4.58 and 4.28, respectively. The variation of the QoE values obtained from testbed measurements in *static* mode is negligible. This is due to the fact the provided bandwidth allows the client to smoothly stream on the lowest quality. It rarely selects segments on the second quality level or suffers from stalling. On average, a MOS of 4.58 is achieved. If the bandwidth mode *trace* is applied



(a) Low level of randomness. **(b)** Increased level of randomness.

Figure 3.14: CDF of the QoE values resulting from the different bandwidth modes with an average bandwidth of 2000 kbps. Black solid lines indicate the HAS performance model's outcome on distributions. The remaining solid lines indicate QoE values computed by the HAS performance model relying on time series as input. Dashed lines represent the results from measurements in the testbed.

during testbed measurements (tb), periods occur where the available bandwidth is not sufficient to download the segment timely, but also periods where the available bandwidth is sufficient to build up a buffer that allows higher quality. Hence, the MOS shows a slight variability between 4.2 and 4.6. Using the proposed methodology relying on the HAS performance model and the Monte-Carlo approach, as indicated by the black solid line, the MOS score is under-estimated for about 70% of the generated video sequences with a high discrepancy compared to the testbed measurement results.

These discrepancies between the measurement results and the generated video sequences decrease with an increasing level of randomness, as shown in Figure 3.13b. Although the QoE scores obtained with the proposed methodology (black solid line) still tend to be lower, the absolute difference compared to the measurement outputs is smaller.

Finally, we discuss the evaluation results when considering an average available bandwidth of 2000 kbps using Figure 3.14. As depicted in Figure 3.14a, the MOS scores obtained with the performance model and fixed inputs are nearly equal for *static* and *trace* with values of 4.585 and 4.584, respectively. It is hence in line with the MOS scores obtained during the testbed measurements with *static* bandwidth, where an average MOS of 4.58 is achieved. The testbed measurements using mode *trace* yield values ranging from 4.04 to 4.58. Again, the majority of generated video sessions under-estimate the MOS values obtained from the measurements.

Figure 3.14b shows that when introducing more randomness in the testbed measurements, the obtained MOS scores diverge more towards to the MOS scores obtained with the analytical model combined with the Monte-Carlo approach. The maximum deviation is less than an absolute score of 0.5.

Finally, we give a brief summary on the results described above. The MOS scores obtained when applying the analytical HAS performance model on distributions and generating video sequences from the probabilistic outputs, show higher variances than the testbed measurements or applying the analytical model with fixed time series input. This is due to the higher abstraction level of the model input parameters and due to the fact that the model might cover more diverse cases. This also includes distinct edge cases, such as the unfavorable combination of having a relative low bandwidth capacity during the download of a comparably large video segment. The probability of such a scenario is lower for the testbed measurements, especially when the level of randomness is low, i.e., no randomization of the bandwidth trace. For that reason, we observe a better approximation of the analytical model's results, when compared to the testbed measurements with an increased level of randomness. In general, the proposed methodology yields a good reflection of the measurement-based results in low bandwidth scenarios. As there are only small discrepancies and as the median QoE scores from the analytical model and the testbed match, the model can be used to reliably approximate the QoE. The analytical model tends to underestimate the QoE in higher bandwidth scenarios, due to the reasons outlined above. However, in this case where the QoE is in general higher and less critical, the model can still be used to obtain a lower-bound or worst-case approximation.

3.5 Use-Cases for Practical Application of the Model

In the following, we practically apply the buffer-based version of the proposed model in two different use-cases. The first use-case is a parameter study to examine the impact of buffer-based quality switching thresholds on the relevant QoE-IFs. The second one addresses possible benefits of using segments with variable durations, an approach which can reduce the video bitrate, but simultaneously increases the variability of the segments' sizes.

3.5.1 Parameter Study on Quality Switching Thresholds

In the following, we exemplary illustrate the model's applicability for parameter optimizations by studying the impact of the quality switching threshold on relevant QoE influence factors. We consider a video split into segments of 5 seconds duration with three different quality levels, hence requiring three quality switching thresholds: qt_1, which is set to 0, analogues to previous parts of this chapter. The threshold for selecting the quality level 2, i.e., qt_2, is subject of the parameter study. Hence, we test four different configurations, namely $qt_2 = \{6, 10, 14, 18\}$. Finally, the threshold for selecting quality level 3, i.e., the highest quality available, is fixed with $qt_3 = 25$.

For the video bitrates of the three quality levels, it holds $q_1 = 0.7 \cdot q_2$ and $q_3 = 1.3 \cdot q_2$, whereby the average bitrate of quality level 2 is 5000 $kbps$ with a standard deviation of 500 $kbps$. We set the bandwidth provisioning factor to $a = 1.5$, i.e., the available bandwidth

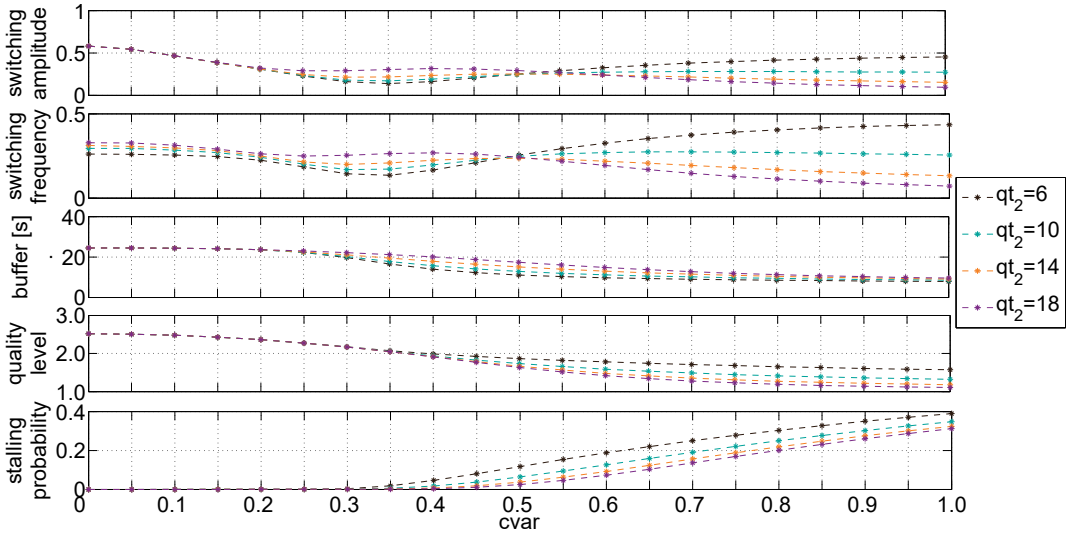


Figure 3.15: Impact of the quality switching threshold qt_2 on QoE-IFs for different coefficients of variation of the available bandwidth (*cvar*). All metrics along the y-axis denote the average value.

is the 1.5-fold of the lowest quality's bitrate. The coefficient of variation of the available bandwidth (*cvar*) ranges from 0 to 1 in steps of 0.05.

The plots in Figure 3.15 illustrate, from top to bottom, the average values for the amplitude of quality switches, the frequency of quality changes, the video buffer level, the quality level, and the stalling probability. For $cvar = \{0, 0.05, 0.1, 0.15, 0.2\}$, the behavior is similar for all configurations of qt_2. For *cvar* ranging between 0.25 and 0.5, the average buffer values start to drift apart, whereby a higher threshold qt_2 indicates a larger buffer. Within this region (*cvar* between 0.25 and 0.5), it is also observable that $qt_2 = 6$ shows the lowest switching frequency and amplitude, whereby $qt_2 = 18$ shows the highest values for these metrics. This is due to the fact that the average buffer, when setting qt_2 to 18 seconds, lies between 22.5 seconds and 17.39 seconds. Hence, the average buffer is close to the switching threshold of $qt_2 = 18$, and as a consequence, quality switches are triggered with higher probability. As the buffer constantly decreases with increasing values of *cvar*, the buffer approaches the values of the lower switching thresholds. As a result, beginning with $cvar = 0.55$, the switching frequency increases with decreasing qt_2.

In general, the buffer shrinks with increasing bandwidth variations. Accordingly, the probability for stalling increases, especially in cases where quality is adapted in a rather aggressive ($qt_2 = 6$), than a conservative manner ($qt_2 = 18$). Although small quality thresholds bring a high quality on average, they should be avoided if the network is likely to show high variability. The results point out that higher values for threshold qt_2 can cushion network dynamics and lead to less stalling, while they provide a similar quality as the lower thresholds for qt_2 in constant scenarios, i.e., when $cvar = 0$.

The threshold qt_2 determines the number of video interruptions in networks with high variability, but at the same time has hardly an impact on the average quality in scenarios with

static network conditions. Accordingly, it is in general better to set the threshold for leaving the lowest quality level to a larger value and thus increase the quality in a more conservative manner.

This first proof-of-concept study shows how the model can be used to optimize quality switching thresholds for a given set of quality levels in a QoE-centric manner. It is essential to adjust configuration-specific parameters, e.g., buffer boundaries for pausing/resuming segment requests or quality thresholds, to the expected network conditions as well as to the given content properties, such as the number of available qualities or their bitrate characteristics. With the proposed generic model, which supports any distributions and configurations, parameters can be tuned to the underlying conditions without the need for costly measurements or simulations.

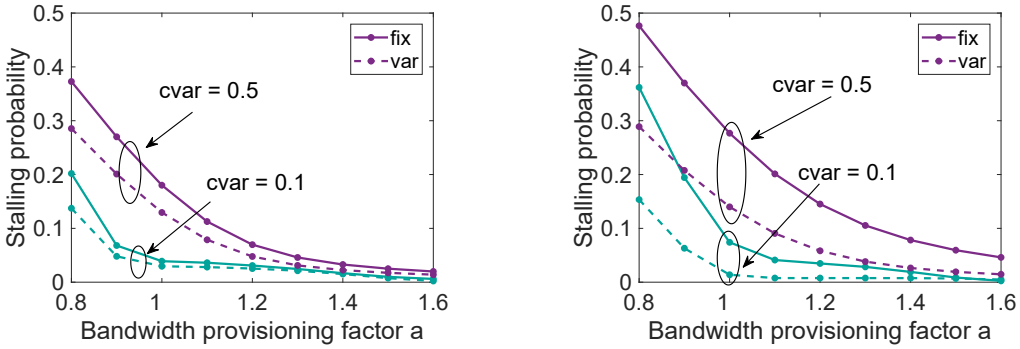
3.5.2 Impact of Variable Segment Durations on Stalling Probability

With HAS, each video segment has to start with an I-frame, a frame containing the whole image information. B- and P-frames are cheaper and only encode the differences to the I-frames. Besides segment starts, I-frames are also included for scene-cuts. Aligning the segment split points with the existing scene-cuts reduces the number of I-frame, and consequently the required bitrate. However, there is a higher variability in terms of the segments' sizes. For instance, while segments of equal durations have similar sizes, the sizes of variable segments are more differing, the larger their difference is in terms of duration. Hence, it is unclear whether the reduced bitrate, but increased segment size variability, is beneficial for the QoE of HAS. Accordingly, we now apply the buffer-based model to study the impact of using video segments of variable durations on the video stalling probability.

As an exemplary test video sequence, we use again the Big Buck Bunny video clip. The video is encoded using a constant rate factor (CRF)[3] of 26. Setting a pre-defined CRF value allows the encoder to dynamically set the bitrate, so to obtain the targeted quality. This means that - independent of the used segment duration - the encoder will retrieve videos of the same quality, but with potentially different bitrates needed to achieve the target visual quality.

We once encode and split the video with a fixed segment duration of 4 seconds each and once with a fixed duration of 10 seconds each. Then, we encode and split the video again, but the encoder can freely choose the segment length up to a fixed maximum duration. This is done two times, once setting the maximum segment duration to 4 seconds, and once setting the maximum duration to 10 seconds. For instance, the durations of the segments are variable, but never exceed the segment durations of the respective fixed version. In the case of 4 seconds (maximum) segment duration, we can economize 5.6% of bitrate with the variable approach compared to the fixed one. In the case of 10 seconds (maximum) segment duration, the variable approach leads to a bitrate reduction of 4.2%.

[3] https://slhck.info/video/2017/02/24/crf-guide.html



(a) 4 seconds.

(b) 10 seconds.

Figure 3.16: Stalling probability of the variable segmentation (var) and the fixed segmentation (fix) approach compared for the Big Buck Bunny clip encoded with a CRF of 26.

For the streaming measurements, we set the the bandwidth provisioning factor as the $\{0.8 : 0.1 : 1.6\}$-fold of the average bitrate of the fixed version, i.e., the videos with a fixed segment duration of 4s or 10s. For the coefficient of variation *cvar* of the available bandwidth, we consider values of 0.1 and 0.5. The model's buffer thresholds p and q are set to 30 and 40, respectively. We omit the quality switching behavior to focus on the segment duration impact on the stalling behavior.

Figure 3.16 illustrates the obtained stalling probabilities. The x-axes denote the bandwidth provisioning factor a, the y-axes represent the stalling probability. As shown in Figure 3.16a, for any of the tested combinations for a and c_{var}, the stalling probability is reduced by the variable segmentation technique. For example, for $a = 0.8$ and $cvar = 0.5$, the stalling probability can be reduced from 0.37 to 0.27, indicating a potential significant improvement of the QoE. Figure 3.16b shows the results for segments with a (maximum) duration of 10 seconds. Again, for any of the tested parameter settings, the variably segmented video leads to a reduction of the stalling probability compared to the respective fixed version. The most significant reduction can be achieved for $a = 0.8$ and $cvar = 0.5$, where the stalling probability can be reduced by 0.18 with the variable approach.

To summarize, the results indicate a high potential of the variable segmentation technique to improve the HAS QoE. However, we want to note here that further effects, such as a possible visual degradation due to the reduction of I-frames, have been neglected. Furthermore, the HAS typical quality switching behavior is not considered. Hence, further studies are needed to investigate in how far adaptation heuristics can cope with segments of variable durations, which will be covered in the subsequent chapter.

3.6 Lessons Learned

This chapter proposed an analytical model based on queueing-theory, which allows to efficiently compute relevant QoE-IFs for HAS. It is able to reflect both, buffer-based as well as rate-based quality adaptation behavior. Despite several assumptions that have been made, the validation showed that for both options, the video buffer can be modeled very accurately. The same holds for the investigated QoE-IFs, whereas the output of the model for the stalling duration in some cases deviates significantly from those values obtained via measurements.

In an initial study we investigated whether the model, despite its inability of considering temporal behavior, is capable of computing the video QoE using time dependent QoE models like P.1203. To do so, we designed an approach based on Monte-Carlo simulations for generating video play back sequences, which in turn can be used as an input for state-of-the art QoE models. The evaluation results show that the output of the abstract analytical model, although it yields generalized results and does not capture the temporal playback behavior, still allows a good estimation of the QoE. This indicates that analytical models may provide a scalable solution for a QoE-aware optimization of parameter settings, such as video segment durations or quality switching thresholds, to the given network conditions.

To demonstrate the practical applicability of the model, we first studied the interplay of switching thresholds and network conditions in terms of the various QoE-IFs. The evaluations show the importance of calibrating the switching thresholds to the network's variability and reveal specific best-practices for setting these thresholds. For example, the threshold for switching from the lowest to the second lowest quality level should always be set in a conservative manner, i.e., the buffer should be sufficiently filled. While a conservative threshold is not harmful in terms of the delivered video quality as long as the network throughput is stable, it can clearly reduce the stalling probability for highly varying network conditions. Secondly, we practically applied the analytical model to evaluate possible benefits of using variable segment durations for HAS. Our evaluations have shown that this approach, due to its capability of reducing the required video bitrate, significantly reduces the stalling probability. Hence, variable segment duration potentially deliver a better QoE to the user, whilst constituting only a minor change to the HAS ecosystem.

4

Variable Segment Durations for Adaptive Video Streaming

The evaluations carried out in the previous chapter indicate that using variable segment durations for HAS potentially reduces the probability for video stallings. These findings obtained via analytical modeling, however, neglect the impact of the ABR behavior and are based on a single video sequence. Furthermore, the conducted study is limited to the impact on video stallings, without considering any other QoE-relevant performance metric. The goal of this chapter is to derive more generalizable conclusions about the benefits of using variable segment durations by evaluating their impact on the coding efficiency and the streaming performance, using a broad set of encoded video sequences and testbed-generated streaming sessions.

The state-of-the-art video segmentation approach relying on fixed segment durations is applied for practical reasons, as it reduces the degrees of freedom of both, the encoding as well as the streaming process. It allows for a fast video encoding, because there are less dependencies the encoder has to consider. In terms of video streaming, the prediction complexity for the ABR algorithm is reduced, since it can rely on a fixed increase of the buffered playtime after a segment download has finished. Fixed segment durations, however, also introduce additional overhead, since a keyframe (I-frame) has to be inserted in a content-agnostic manner at the beginning of each segment to allow its independent playback.

Technically, the HAS principle permits utilizing segments of variable durations, as long as the segment start times, i.e., the I-frame locations, are consistent between the different

quality levels. Variable segment durations are the result from a segmentation technique, which takes the video characteristics into account. For instance, aligning the segment split positions with I-frames that are needed a-priori, e.g., due to scene-cuts, results in segments of different lengths. Netflix refers to this approach as shot-based encoding [113], and the previous chapter of this monograph already provided an initial insight of the potential to reduce the stalling probability resulting from the increased encoding efficiency. However, a large-scale comparative analysis, studying impact factors such as the compression rate, video resolution, or the magnitude of segment duration variability, has not been conducted yet. Besides, while few existing works [114, 115] study the gain in terms of compression efficiency, these works still neglect the influence of variable segment durations on the video streaming process itself. As an increasing variability of the segment durations results in an increase of the segment size variations, it affects the decisions of ABR algorithms and therewith the streaming performance [65]. Hence, to show the applicability and potentials of variable segment durations for today's HAS systems, it is essential to conduct a broad investigation of their influence on the streaming behavior, which also does not exist so far.

This gap is addressed in the course of this chapter. In order to evaluate and compare the fixed and the variable approach with regard to encoding-related metrics – such as segment durations, file size, or video quality – we create a large data set consisting of roughly 2,000 encoded video sequences. For the sake of representativeness of our results, we choose four publicly available video sources with durations between 8 and 12 minutes, all with a resolution of up to 2160p. We use the H.264 video compression standard and consider variable bitrate encoding (VBR) as well as constant bitrate encoding (CBR) with different constant rate factor (CRF) settings and target bitrates. Evaluations carried out using this data set show that variable segment durations can reduce the video bitrate by up to 15%, while maintaining a comparable video quality, as shown by the SSIM metric. Furthermore, we reveal the relevant factors influencing the potential for bitrate reduction.

To study the impact of variable segment durations on HAS performance, we run extensive testbed measurements. Thereby, we consider varying network conditions, different state-of-the-art ABR strategies, and a subset of the encoded video sequences, including both, fixed and variably segmented videos. Thus, we obtain more than 7,000 streaming sessions, which are evaluated in terms of their QoE scores according to the model from ITU-T Recommendation P.1203. Our evaluations reveal a slight increase in the median QoE of all streaming sessions when using variable segment durations and significant increase of the subjective streaming performance in scenarios with low bandwidth capacities.

The rest of this chapter is structured as follows. Section 4.1 gives background information and presents related work focusing on the impact of segment durations for HAS. Section 4.2 introduces the proposed variable approach. Section 4.3 details on the investigations carried out related to video encoding and segmentation. Subsequently, the impact of the variable approach on the streaming performance is addressed in Section 4.4. Finally, Section 4.5 summarizes the lessons learned.

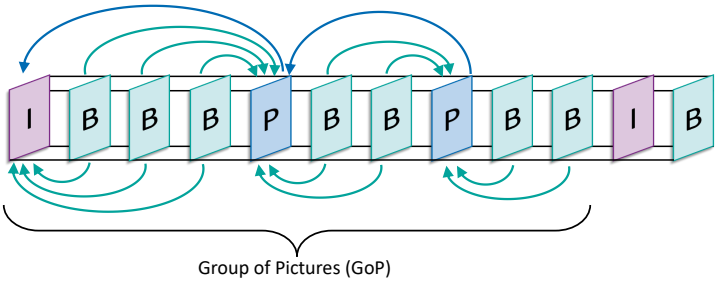


Figure 4.1: Structure and references of different frame types in digital video coding. Arrows denote references between the frames.

4.1 Background and Related Work

This section firstly introduces digital video encoding basics with a focus on the different types of frames that are used throughout the encoding process. Next, we describe the state-of-the art mechanism for preparing HAS content with fixed segment duration. The SSIM metric is introduced subsequently, as this is one of the most prominent metrics to assess video quality and also used in the course of this chapter. Finally, we present scientific studies which focus the impact of segment durations on HAS performance and outline related works on variable segment durations.

4.1.1 Digital Video Encoding

Generally spoken, a video is not more than the concatenation of still images, which - when displayed in chronological order with a rate of at least about 24 images per second - is perceived as motion by humans. Raw, i.e., uncompressed, digital videos indeed do rely on the purely succession of pictures. While they come without any degradation of visual quality, raw videos result in very high file sizes, making them inefficient for storage and transmission. Video encoding techniques allow to store the video information in a more efficient manner by exploiting the temporal redundancy of subsequent images. For instance, as long as no scene-cut occurs, the color and location of the pixels of an image i are very similar to those of an image $i+1$.

Digital video encoding makes use of the high correlation of pixels of successive images by only considering the information about the changes between two subsequent images. The movements of pixels are encoded in so-called motion vectors, which are substantially cheaper to encode than storing the whole image information. In this context, different types of video frames need to be taken into account, which differ with respect to the information they contain and in terms of the frames they are referring to.

Figure 4.1 exemplarily illustrates an encoded video scene using I-, P-, and B-frames, as applied, e.g., with the H.264 video compression standard. The Intra-coded picture, or I-frame, contains the complete picture information and is basically like any other image, such as JPEG. An I-frame does not require any other frame to be decoded, but is simultaneously

the most expensive type of frames, i.e., it has typically a comparably high file size. P-frames (Predictive-coded picture) refer to previous I-frames or other previous P-frames. More specifically, a P-frame encodes the difference compared to the previous frame it refers to. For example, if a person is moving through a static nature scene, only the person's movement needs to be encoded, while no information regarding the unchanging background has to be stored. As a consequence, P-frames are more compressible than I-frames and their usage increases the encoding efficiency. Finally, there are B-frames (Bidirectional predicted picture). Compared to P-frames, which only refer to previous frames, B-frames can have preceding as well as subsequent frames as their reference. Due to their option to refer to two or more frames, B-frames are the most efficient ones in reducing the size of the frame, while maintaining the visual quality. The sequence of one I-frame and all following frames which refer to that I-frame is denoted as a Group of Picture (GOP) and each GOP of an encoded video sequence can be decoded independently.

4.1.2 State-of-the-Art Video Preparation for HAS

As they are complete images, I-frames typically have a significant larger file size than P- or bidirectionally B-frames, which only encode the differences compared to a reference frame. Hence, I-frames should be used sparingly as refresh points for the decoder. At scene cuts, however, the placement of I-frames can yield lower file sizes, as predicting from a previous picture would be less efficient, that is, it would require more bits to code the difference than to simply create another I-frame.

With HAS, all encoded video segments must be playable independently to allow for the quality adaptation with each segment. This requires them to start with an I-frame – more specifically, an instantaneous decoder refresh (IDR frame) – which is inserted during the segmentation process. Typically, it is recommended to encode videos for HAS using strictly fixed I-frame intervals. Depending on the used technology and intended encoding latency, these intervals range from 2 to 10 seconds [46, 116]. Choosing fixed intervals has practical reasons, since scene-cut detection can be disabled, and the encoder can work in a "set and forget" mechanism. This approach, however, lowers the encoding efficiency, as more I-frames are needed, particularly if a very short segment duration is chosen. Aligning the segment durations with existing I-frames – which are needed anyway due to scene-cuts – could reduce this overhead. This, however, would result in video segments that have different durations. While from a technical point of view, variable segment durations can be used for HAS, there are some practical challenges associated with this method, as we will describe in more detail in the course of this chapter.

4.1.3 Structural Similarity Metric

The structural similarity (SSIM) metric by Wang et al. [72] is widely used for quantifying the visual quality of a video. It is an FR metric, which means that an uncompressed

distortion-free frame is used as reference and compared with another, potentially visually degraded frame. The SSIM takes into account three intermediary scores, which express the differences in terms of luminance, contrast, and structure. They are typically computed in a sliding window fashion on small pixel blocks, e.g., of size 8x8 [117], moving horizontally and vertically from the top left to the bottom right pixel of a frame and are later composed to one final score. The windows within the reference and the distorted frame are supposed to be non-negative image signals and are denoted as x and y in the following.

$$luminance\ l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

Thereby, μ_x and μ_y are the averages for the images x and y and C_1 denotes a constant which avoids instability when $\mu_x^2 + \mu_y^2$ is nearly zero.

$$contrast\ c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Again, a constant factor, in this case C_2, is introduced to account for $\sigma_x^2 + \sigma_y^2$ being nearly zero. The terms σ_x and σ_y denote the standard deviations.

$$structure\ s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

The term σ_{xy} denotes the cross-variance and similar as above, C_3 stabilizes the division. The three components are combined as follows to obtain the final similarity index for images x and y:

$$SSIM(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma$$

The parameters $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ allow to put different weights to the three components. If luminance, contrast, and structure should be equally and fully considered, i.e., $\alpha = \beta = \gamma = 1$, the final SSIM score can be denoted as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

As this formula yields the SSIM score for a single frame, it needs to be successively applied to each single frame of a video. When denoting a video's SSIM score throughout this chapter, we refer to the average SSIM score of all its frames. The SSIM values range from 0 to 1, including both borders. An SSIM value of 1 means equality between two frames. Consequently, the higher the SSIM value, the lower the visual degradation due to compression. In contrast to other metrics used for image quality assessment, like for example peak signal-to-noise ratio (PSNR), SSIM is capable to consider the subjective quality by combining the three aforementioned factors and is thus well known for its high correlation with the human perception of degradations [118].

4.1.4 Studies on Segment Durations for HAS

The impact of segment durations has been focus of several studies, as it is a crucial factor for the performance of adaptive video streaming and for the user's satisfaction with the service [119]. Shorter video segments allow a more fine-granular adaptation of the video quality to current network conditions. However, short video segments decrease the encoding efficiency [120] and increase the signaling overhead, as each segment download is initiated via a dedicated HTTP request.

In order to analyze the performance of HAS for different fixed segment durations in a mobile environment, [121] applies flow-level modeling. The developed mathematical models show that using segments of shorter duration should be used, as they allow higher frequencies for quality adaptations and yield better properties in terms of video playback smoothness. To overcome the issue of increased signaling overhead, the authors propose to request several video segments at once. Liu et al. [122] examine how to set the segment duration so as to optimize the client's TCP throughput estimation, which in turn allows for an optimized bitrate adaptation when rate-based heuristics are used. They conclude that the segment duration should be set as the minimum duration in which the average reception rate can represent the end-to-end networks capacity.

The selection of segment durations in live-streaming scenarios is discussed in [59]. The paper evaluates the trade-off between the high responsiveness of short segments in terms of quality adaptation versus their increased encoding overhead when selecting a suitable segment duration. The authors show that streaming with segments of a sub-second duration allows to reduce the start-up delay and camera-to-display delay. By using the HTTP/2 push feature, they overcome the issue of increased signaling overhead. Depending on the network RTT and the segment durations, the authors derive an optimized number of k segments to be pushed to the client with a single HTTP GET request, so to optimize the streaming performance.

The work presented in [123] proposes to use different segment durations, depending on the current HAS phase. The HAS videos are segmented multiple times so to obtain several representations in terms of the segment duration. During the start-up phase, the client requests very short video segments. With increasing buffer size, longer segments are fetched

from the server. Using this method, the authors combine the advantages of a low start-up delay (due to short segments) with the advantage of increased encoding efficiency resulting from longer segments during a steady playout phase. The idea of providing the video content not only in different qualities, but additionally split in segments of different durations is also presented in [124]. The idea behind is to use longer segment durations on higher quality levels to make use of the high compression efficiency and to use the shorter duration representations for lower bitrates to achieve a faster quality adaptation in case of sudden bandwidth fluctuations. Accordingly, an ABR algorithm is proposed which does not only choose the next segment's bitrate, but also the segment's duration. A weakness of this approach is the constrained possibility for switching between representations of longer and shorter segment durations, as this is only possible where the segments' starting points are synchronized.

Although the works above consider several representations with different segment durations, the durations within these representations are still fixed. The idea to improve the alignment of the video segments with the video content has initially been proposed and compared with fixed video segments in [115], which found on a small that of encoded videos that on average 10% of bitrate can be saved. In 2018, Netflix proposed shot-based encoding in their *Dynamic Optimization* approach, which utilizes variable segment durations and thereby allows for improved rate-distortion optimizations for each shot [113]. Similarly, [114] investigates the impact of the variable approach on the resulting segment sizes and their variability. Thereby, the authors consider among others different video categories, resolutions, and video codecs.

4.2 Variable Segment Durations for Adaptive Streaming

The state-of-the art mechanism for preparing HAS content relies on fixed segment durations, which increases the encoding overhead, and thus the bitrate requirements due to additionally inserted I-frames, which are comparably expensive. In the following we explain how the proposed variable approach can overcome this issue based on a real video example and a detailed focus on the frame structure during video encoding and segmentation. Some best practices and necessary player capabilities should be kept in mind when using segments of variable duration, to avoid drawbacks during streaming. We shortly refer to them at the end of this subsection.

4.2.1 Variable versus Fixed Segment Durations

Figure 4.2 illustrates the first 45 seconds of the *Big Buck Bunny* video with 24 frames per second. It is used in the following to discuss the differences between the fixed and variable segmentation approach. More specifically, we focus on the procedure of splitting the video and placing the segment start positions in a content-depending manner. In this example, we assume a maximum segment duration of 10 seconds (240 frames per segment) for the

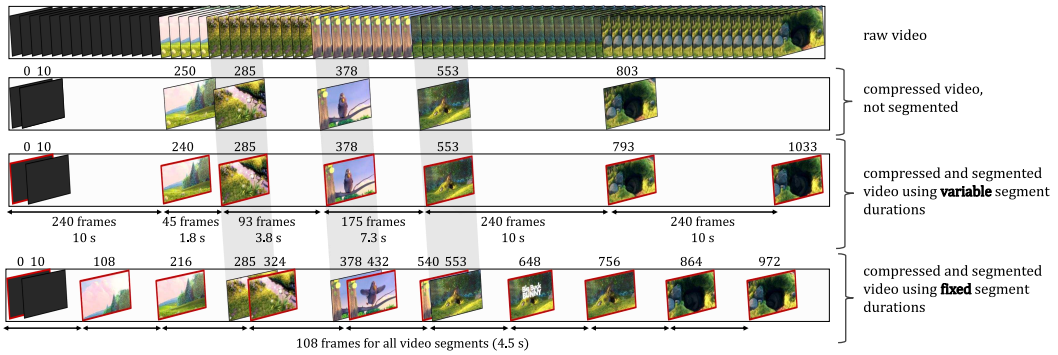


Figure 4.2: Exemplary illustration of the I-frame placement for unsegmented videos and segmented videos with the fixed and the variable approach. The first line denotes the raw video, the second line represents the encoded, but not segmented version of the video. The two bottom lines show a segmentation using fixed durations and variable durations, respectively. Red frames indicate I-frames at the beginning of a video segment.

variable approach. The setting of a maximum duration is done in order to avoid that they become too long. For the fixed approach, we set a duration of 4.5 seconds, i.e., 108 frames per segment.[1]

The top box of the figure represents the raw video, where each frame contains the complete image information. The second box illustrates a compressed, but not segmented video. I-frames are inserted when the scene changes, which happens at frames 0, 10, 250, 285, 378, 553, and 803. For the remaining part of the video, the encoder relies on cheaper P- and B-frames, which are omitted in the illustration for the sake clarity.

The third box illustrates a segmented video when using the proposed variable approach. Similar to the unsegmented video, frames 0 and 10 are I-frames. At frame 240, the maximum duration of 10 seconds is reached for the first segment and a new segment has to start. Consequently, frame 240 must be encoded as an I-frame. This I-frame (240) can then also be used to account for the scene-cut, which is captured in the unsegmented sequence with the I-frame at position 250. Hence, with the variable approach, frame 250 can be encoded more efficiently by using a P- or B-frame. This is possible because the encoder has a certain degree of freedom in terms of where to place an I-frame for an efficient encoding and segmentation. This especially holds when a scene does not change abruptly, but instead with fading effects. The next three segment beginnings are aligned with the scene-cuts. For instance, the existing I-frames of the unsegmented video, that is, frame 285, 378, and 553, can be used as segment start points. Finally, segments are split at frame 793 and frame 1033, because the maximum segment duration limit of 240 frames is reached.

The bottom box depicts the fixed segmentation, where all segments must have a duration of 4.5 seconds. Two of the frames (250, 803), which used to be I-frames in the unsegmented

[1] Later parts of this chapter will show that a maximum segment duration of 10 seconds with the variable approach results in an average segment duration of 4.5 seconds. This means that the illustrated values result in the same number of video segments for the fixed and the variable approach, when considering the entire video.

video, can be replaced by a cheaper frame-type, because an I-frame was inserted nearby. However, due to the strictly fixed segment duration of 108 frames, I-frames are placed at 324 and 432, despite the small differences to their preceding I-frames, which are required because the scene changed. Another I-frame, inserted for the sake of a constant segment duration, is 540. Roughly half a second later, i.e., after 13 more frames, an I-frame is nevertheless needed as the scene changes again. The total number of additionally needed I-frames, due to video segmentation, sums up to 7 in the fixed case, while only 1 additional I-frame is needed in the variable case for the illustrated sequence of 45 seconds.

4.2.2 Requirements and Best Practices

The HAS principle of adapting the video quality prior to a segment's download does not hinder the usage of variable segment durations. Nevertheless, some requirements need to be fulfilled in order to implement this approach in a real system: Firstly, the segment boundaries have to be aligned along all available quality representations of one video, that is, video bitrates and resolutions. Otherwise, even the slightest deviation will provoke a skip or a repetition of frames at quality switches, which may impair the user's experience.

Secondly, the player implementation must be agnostic to changing segment durations, that is, it has to consider each segment's duration individually. During our tests, we found that some player implementations assume fixed segment durations per se. The TAPAS player [110], which is intentionally kept simple to ease the integration of own heuristics, only captures the duration of the first segment and works on the premise that all other segments have exactly this duration. This would result in wrong buffer computations and consequently lead to wrong decisions carried out by the ABR strategy. Another similar issue was found with the `dash.js` reference player in version 2.9.3. When the *insufficient buffer rule*[2] rule was triggered, it mapped the current segment's duration to the 10 subsequent segments when estimating their download duration. As this estimation was based on the wrong segment duration, the player over- or under-estimated the download duration, resulting in a too optimistic or too pessimistic behavior in terms of quality selection. The issue has been addressed with version 3 of `dash.js`.

Finally, a practical maximum segment duration should be defined. The longer a segment's duration becomes, the longer it takes to download it. If the download duration exceeds the buffered time, it will cause video stalling. Besides, the definition of a maximum segment duration caps the variability of the segment sizes and furthermore guarantees that an I-frame is placed after a certain time interval, which maintains the visual quality on the one hand and the adaptability of the video stream on the other hand.

[2] `https://github.com/Dash-Industry-Forum/dash.js/wiki/ABR-Logic`

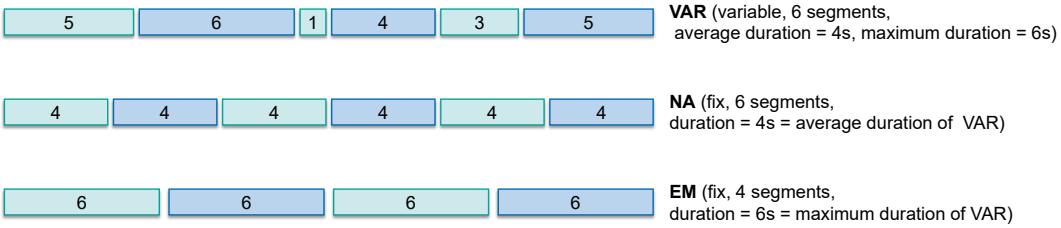


Figure 4.3: Exemplary illustration of the two fixed segmentation options *NA* and *EM* used as a comparison against the variable approach (*VAR*). The bars denote the video segments along with their duration in seconds.

4.3 Impact on Video Encoding Efficiency

First of all, we quantify potential advantages of the variable approach compared to the fixed segmentation with respect to the encoding efficiency. That is, we study the bitrate reduction and evaluate in how far the visual quality of the video is impacted by the reduction of I-frames. Therefore, we use four source videos, which are encoded and segmented with a multitude of different parameters, leading to a large data set of video sequences. This allows us to reveal the relevant factors, which impact the potential of the variable approach. We present in the following the applied methodology and describe the obtained results afterwards.

4.3.1 Evaluation Methodology

This subsection introduces the methodology for studying the video encoding and segmentation process with the variable and the fixed approach. First of all, we introduce the terminology used throughout this chapter in order to fairly compare both approaches. Then, we present the four source videos used to create the data set of encoded sequences, along with their most important characteristics. In a next step, we describe the video encoding and segmentation process along with the different parameter options, as well as the (maximum) segment duration settings. Finally, we provide insights to our encoding architecture, which is used to generate the data set.

4.3.1.1 Terminology

When using the variable approach, we define a maximum duration (*max_dur*) for the video segments. The encoder can freely choose a segment's duration within the range from 0 to *max_dur* seconds. Please note that the definition of a sensible maximum duration is important to avoid segments of too long duration, as they would reduce the adaptability of the video stream. Throughout subsequent parts of this chapter, we use two methods to compare the performance of the variable and the fixed approach, as illustrated in Figure 4.3.

The first line represents a variably segmented video, where the encoder can freely choose the segment duration within a range of 0 to 6 seconds. The illustrative example yields an average segment duration of 4 seconds and 6 segments in total. The second line shows the first option to compare the variable and the fixed approach, where the fixed-segment sequences are evaluated against those variable-segment sequences, which have (nearly) the same *average* segment duration, with a granularity of half a second. For instance, if the variable approach with $max\_dur = 6$ seconds yields an average segment duration of 4.3 seconds, this is compared to the fixed segmentation with a duration of 4.5 seconds. If it yields an average segment duration of 3.9 seconds, this is compared to the fixed segment duration of 4.0 seconds. We refer to this option as *nearest average (NA)*. Finally, the third line illustrates the second comparison option. It evaluates the variable-segment encodes against those fixed-segment encodes, which have the same duration as the specified *max_dur*. In the shown example, the variable approach with $max\_dur = 6$ seconds is compared to the fixed approach with a segment duration of 6 seconds. Accordingly, we refer to this approach as *equal max (EM)*.

The *NA* comparison yields (nearly) the same number of video segments, and hence results in the same signaling overhead when it comes to video streaming. With *EM*, the total number of segments is lower for the fixed duration video. As a consequence, an *EM* video can be streamed with lower signaling overhead than the respective *VAR* video. While the increased signaling overhead might be disadvantageous for *VAR* on the one hand, it is beneficial on the other hand in the sense that the quality can be adapted more frequently, due to the comparably shorter segment durations. In the remainder of this chapter, we refer to the approach applying variable segment durations as *VAR*, and we use *NA*/*EM* for the fixed-duration segmentation.

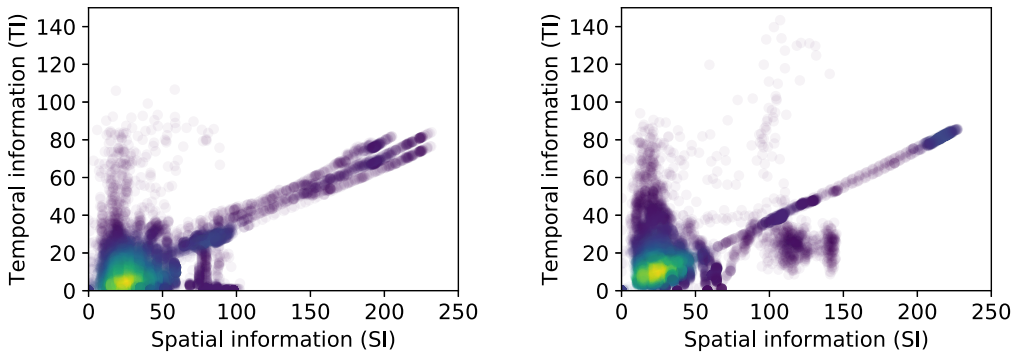
4.3.1.2 Source Videos

Our evaluations are conducted based on four freely available videos. The clips *Big Buck Bunny (BBB)*, *El Fuente (ELF)*, and *Tears Of Steel (TOS)* are provided by the Blender foundation[3], while *Meridian (MER)*[4] is a test sequence from Netflix. The sources are scaled from their original dimensions to 2160p, 1080p, 720p, 480p, and 240p resolution (using bicubic filtering), with 24 frames per second (using `ffmpeg`'s `fps` filter).

Spatial and temporal information (cf. ITU-T Rec. P.910) are widely used as an approximate for video complexity. The spatial information (SI) is computed from a single frame and estimates the image complexity by computing the standard deviation of each pixel in a filtered version of the image. The temporal information (TI) is computed as the standard deviation of the motion difference between two consecutive frames. The higher the SI and TI values for a given video, the higher is its complexity.

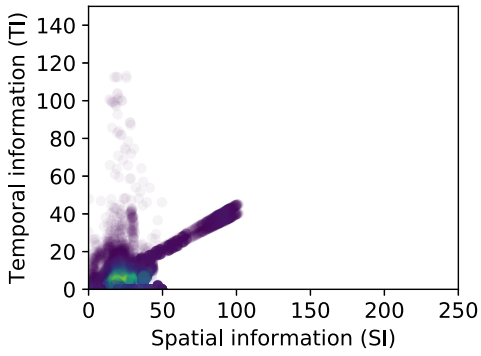
[3] `https://www.blender.org/`

[4] `https://medium.com/netflix-techblog/engineers-making-movies-aka-open-source-test-content-f21363ea3781`

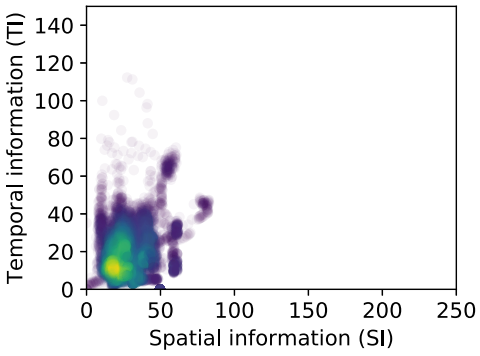


(a) Big Buck Bunny.

(b) Tears Of Steel.



(c) Meridian.



(d) El Fuente.

Figure 4.4: Spatial and temporal information of the four source videos. Lighter areas indicate higher density, darker areas lower density, respectively.

Table 4.1: Characteristics of the considered source videos.

Full Name	Abbreviation	Mean SI	Mean TI	Duration	Category
Big Buck Bunny	BBB	49.923	17.575	10:34	Cartoon
El Fuente	ELF	28.994	20.194	07:57	Documentary
Meridian	MER	28.541	8.732	11:58	Mystery
Tears of Steel	TOS	45.307	21.027	12:14	Action

The ranges for the spatial and temporal complexity of the videos in our test set are illustrated in Figure 4.4. The x-axes denote the SI, the y-axes the TI, respectively. Lighter areas indicate higher density, darker areas indicate lower density, respectively. Table 4.1 further denotes the corresponding average values along with additional video characteristics. Besides the varying spatio-temporal complexity, these videos have been chosen due to their different categories, as well as for their durations of at least about eight minutes, to support meaningful conclusions.

4.3.1.3 Encoding Methods

For encoding the videos, we use `ffmpeg` and apply both of the major rate control option used for encoding, which either achieve a target visual quality or a target (nearly constant) bitrate for the encoded bitstreams:

- **Variable Bitrate Encoding (VBR):** one-pass, using the x264 Constant Rate Factor (CRF), which results in a roughly constant quality.[5]
- **Constant Bitrate Encoding (CBR):** two-pass, using a target bitrate (br) and Virtual Buffer Verifier (VBV) constraints of $maxrate = 1.25 \cdot br$ and $bufsize = 2 \cdot br$.[6]

While VBR encoding has a lower variation in visual quality over time, it leads to higher bitrate variations, which may impair the streaming performance. CBR, on the other hand, keeps the bitrate static, within constraints, along the video, resulting in possible quality degradations in scenes that are more spatio-temporally complex. Video streaming services have mainly been relying on CBR-encoded content due to the reduced bitrate variations, but streaming with VBR-has gained more significance over the past years [125, 126, 127].

With VBR, the CRF accounts for the motion in the video to determine the bitrate needed to achieve a certain quality. For our analysis, we consider four different settings for the CRF, i.e., $crf \in \{16, 22, 28, 34\}$. The lower the CRF, the higher is the resulting video quality, whereby a value of 16 can be considered as visually lossless. An increase of the CRF value by 6 will roughly halve the resulting bitrate.[7] To determine the target bitrates for the CBR approach, we first encode each video in all resolutions, as well as using all specified segment durations using VBR encoding with the four different CRF values. For each four-tuple $\{video/resolution/CRF/duration\}$, the average resulting bitrate is then used as the target bitrate for CBR. By doing so, we obtain similar average bitrates for both, VBR and CBR encoded videos.

4.3.1.4 Segment Durations

Video segments of too long durations should be avoided. Otherwise, the quality might be degraded, short-term quality adaptations are not possible, and the increased download time could lead to re-buffering. Hence, we specify an upper bound for the maximum duration of variable segments, denoted as *max_dur*. In order to compare the efficiency of different variable segment lengths, we choose four different settings, i.e., $max\_dur \in \{4, 6, 8, 10\}$ seconds. For the fixed durations, we use the values according to *NA* and *EM* resulting from the variable segmentation.

For the encoding with fixed segment durations, the `ffmpeg` option `force-key-frames` is used, and scene-cut detection is deactivated. In order to determine where to split the

[5] `https://slhck.info/video/2017/02/24/crf-guide.html`
[6] `https://slhck.info/video/2017/03/01/rate-control.html`
[7] `https://trac.ffmpeg.org/wiki/Encode/H.264`

Table 4.2: Parameter settings for video encoding and segmentation.

Characteristic	Values
Source video	BBB, TOS, MER, ELF
Resolution	240p, 480p, 720p, 1080p, 2160p
Frames per second	24
Encoding method	Variable Bitrate (VBR), Constant Bitrate (CBR)
Constant rate factor (CRF) used for VBR	16, 22, 28, 34
Target bitrate used for CBR	Average bitrates resulting from VBR encoding
(Maximum) segment duration	*VAR* and *EM*: 4s, 6s, 8s, 10s *NA*: Average durations resulting from *VAR* (see Table 4.3)

video when using the variable approach with a given maximum duration, we apply the following procedure: For each video, we choose its version in 2160p resolution as the reference. This reference is encoded and segmented with the `force-key-frames` option set to *max_dur*, i.e., keyframes are only forced if no keyframe was inserted since *max_dur* seconds. Furthermore, we do not specify any segment duration with the `seg_duration` option. This allows the encoder to freely choose the segment durations between 0 and *max_dur* seconds. All frame positions, at which the encoder decides to split the video, are logged during the encoding of the reference. These logged positions are then used as an `ffmpeg` input when encoding and segmenting the remaining video representations. This ensures that we split at exactly the same positions along each resolution and target bitrate or quality for a given video. The described procedure turned out to be necessary, as in rare cases, the split positions deviated by a few frames from one resolution to another. This was caused by the scene-cut detection of the encoder, which treats inter-frame differences differently at lower resolutions.

4.3.1.5 Encoding Architecture and Quality Calculation

As outlined in the previous parts, we apply a wide range of different encoding and segmentation options. A summary of all parameter settings used is given in Table 4.2. Each of the 4 videos is encoded in 5 distinct resolutions. We encode each clip once using VBR encoding with 4 different CRF values, and once using CBR encoding with the target bitrate set as the average bitrate resulting from the VBR encoding. We furthermore consider 4 settings for *max_dur* when using *VAR*. In order to obtain all respective *EN* and *NA* encodings, each clip is additionally segmented using 8 fixed durations. Hence, in total our data set consists of $4 \cdot 5 \cdot 4 \cdot 2 \cdot (4+8) = 1{,}920$ encoded video sequences.

The encoding and segmentation process itself, as well as the subsequent evaluation of the resulting videos, is time and resource consuming. In order to support encoding on any platform and to easily distribute the encoding and evaluation process on several computing nodes, the tasks were encapsulated within a Docker[8] container, publicly provided on Github.[9] Each Docker instance obtains one task, which is defined by the source video ID

[8] `https://www.docker.com`
[9] `https://github.com/fg-inet/docker-video-encoding`

and a combination of encoding parameters. These parameters are summarized in a job description that includes the segmentation option (fixed vs. variable), the encoding option (CBR vs. VBR), the (maximum) segment duration, and a target bitrate or CRF value.

When the video segmentation and encoding is completed, the container analyzes the resulting video and determines several parameters. Firstly, it yields encoding quality-relevant metrics on a per-frame basis, such as the SSIM metric and PSNR, calculated against the 2160p source via `ffmpeg`.[10] Secondly, it analyzes the resulting bitrates and retrieves detailed frame characteristics, such as the type and size. Thirdly, it generates a timeline of the video segments, specifying their duration and file size.

4.3.2 Evaluation Results

In the following we evaluate and compare the variable and the fixed segmentation approach with regard to several aspects. Firstly, we investigate the segment durations resulting form the variable approach with different settings for *max_dur*. Secondly, we examine the number of I-frames which can be economized by the proposed approach and quantify the potential bitrate reduction. In a next step, the impact of the variable approach on the visual quality is quantified. Lastly, we study the influence of relevant video parameters, such as the CRF value or the segment duration, on the encoding efficiency gain.

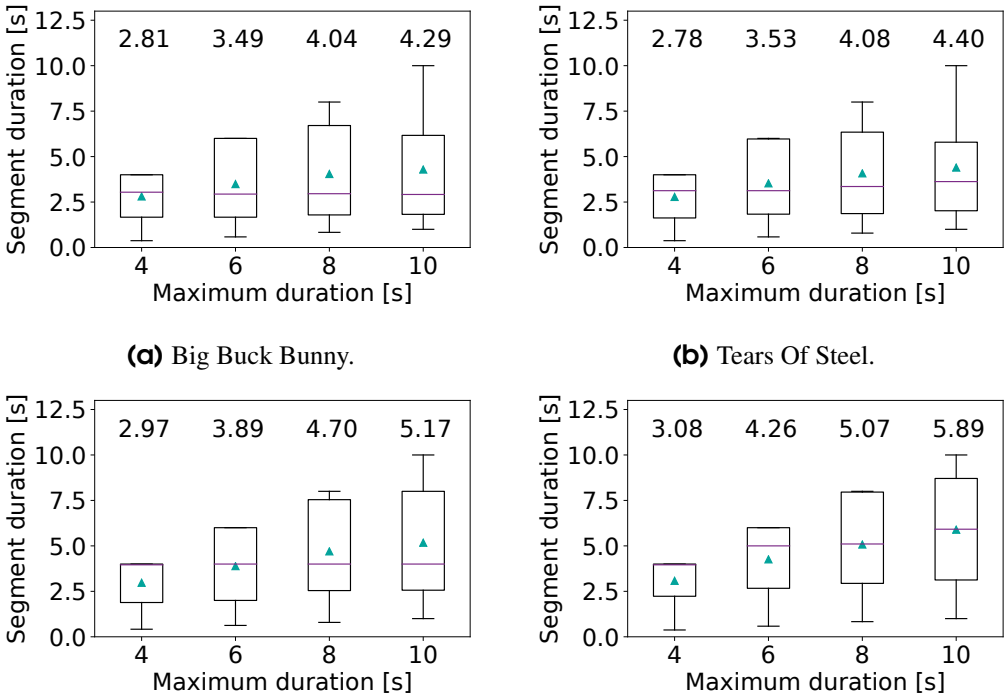
4.3.2.1 Resulting Segment Durations

The box plots shown in Figure 4.5 illustrate the segments' durations resulting from the variable segmentation technique. The x-axis denotes the configured maximum duration (*max_dur*), the y-axis denotes the resulting duration. In the case of Big Buck Bunny (Figure 4.5a), Tears Of Steel (Figure 4.5b), and Meridian (Figure 4.5c), the median segment durations nearly remain static along all *max_dur* settings. However, the encoder makes use of longer segments if admitted, leading to increased average durations for each video clip when increasing the maximum duration setting. With a maximum duration of 4 seconds, the longest average durations can be observed for Meridian and El Fuente (Figure 4.5d), as for these videos, the median duration corresponds almost to the maximum duration.

El Fuente results in the largest average duration among the investigated videos for any of the configured maximum durations. Among all considered video clips, ELF has the lowest number of I-frames relative to its video duration. This indicates that there are fewer scene-cuts, and consequently less possibilities for the encoder to split at I-frames which are anyways needed.

As described above, we compare the variable approach against the fixed either based on the same average duration (*NA*), or based on the same maximum duration (*EM*). Table 4.3 summarizes the information retrieved from Figure 4.5 to allow a quick lookup of the fixed

[10] `https://github.com/slhck/ffmpeg-quality-metrics`



(a) Big Buck Bunny.

(b) Tears Of Steel.

(c) Meridian.

(d) El Fuente.

Figure 4.5: Segment durations resulting from *VAR* with different maximum duration settings. Purple lines denote the median, boxes denote the 25th to 75th percentile. Green markers and the numbers on top indicate the average segment duration denoted in seconds.

Table 4.3: Fixed segment durations according to *EM* and *NA* resulting from *VAR* with different maximum duration settings for the source video clips. All values are denoted in seconds.

	BBB		**TOS**		**MER**		**ELF**	
VAR	*EM*	*NA*	*EM*	*NA*	*EM*	*NA*	*EM*	*NA*
0-4	4	3	4	3	4	3	4	3
0-6	6	3.5	6	3.5	6	4	6	4.5
0-8	8	4	8	4	8	4.5	8	5
0-10	10	4.5	10	4.5	10	5	10	6

segment durations corresponding to a segmentation using *VAR*. Thereby, *VAR* denotes the range of allowed segment durations in the variable case, and *EM* and *NA* denote the respective fixed segment durations, against which *VAR* is compared. Please note that we specify a maximum granularity of half a second for the fixed segment durations. For instance, when determining the *NA* durations we round the average values obtained via *VAR*, which results in a slight deviation of the values shown in Figure 4.5 from the values given in Table 4.3.

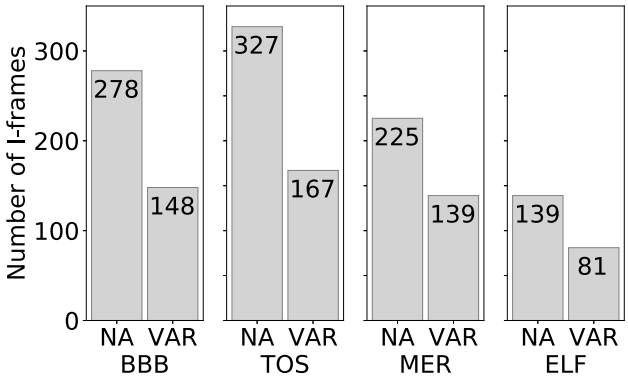


Figure 4.6: Absolute number of I-frames. Comparison of *VAR* using a maximum duration of 10 seconds against the respective *NA* fixed segment durations.

4.3.2.2 Reduction of I-frames

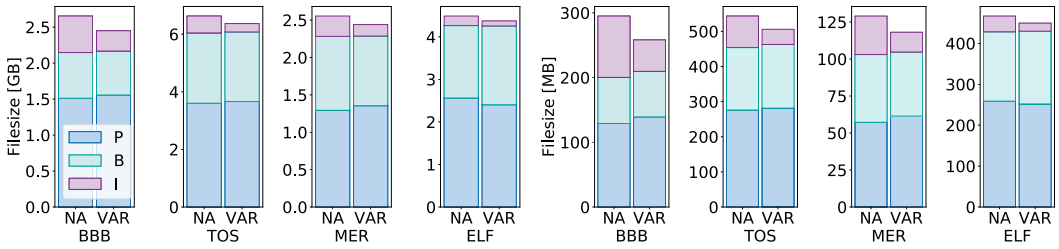
In the following, we investigate the number of I-frames that can be reduced by using the variable approach for the different videos. For these evaluations, we confine on VBR encoding and on the comparison using *NA*. Please note that each video segment strictly must start with an I-frame. Compared to *EM*, *VAR* increases the number of I-frames due to the higher number of resulting video segments, while *NA* yields the same number of segments as *VAR*. Hence, the *NA* is the better option to fairly study the potential I-frame reduction that can be achieved with *VAR*.

Figure 4.6 illustrates the number of I-frames needed when using *VAR* on the example of a maximum segment duration of 10 seconds and the number of I-frames needed when considering the respective *NA* fixed segment duration for the investigated video clips. Note that for the same segment duration setting of one video, the number of I-frames is equal along all resolutions and quality representations, as well as for VBR and CBR encoding. Although both, *VAR* and *NA* segmentation, result in practically the same number of video segments, *VAR* requires a significantly reduced number of I-frames.

For the BBB clip, we obtain 148 I-frames with *VAR*, compared to 278 with *NA*. Hence, the costly I-frames can be reduced by a total of 130. Accordingly, the *VAR* approach can economize 46% of the expensive frames in this case. The relative reduction is similar for the remaining videos. For instance, 49%, 38%, and 42% of the I-frames can be saved for TOS, MER, and ELF, respectively.

4.3.2.3 Reduction of File Size

In a next step, we exemplarily study the impact of the I-frame reduction on the overall video file size. For each of the four video clips, we consider its 720p video resolution, once encoded with a CRF value of 16, i.e., the highest visual quality considered during our evaluations, and once with a CRF value of 34, the lowest quality, respectively. Figure 4.7a



(a) Filesize for 720p and CRF 16.

(b) Filesize for 720p and CRF 34.

Figure 4.7: Overall filesize for VBR encoded videos. Comparison of *VAR* using a maximum duration of 10 seconds against the respective *NA* fixed segment durations.

illustrates by means of different colors the contribution of the three different types of frames to the overall file size for the example of $CRF = 16$.

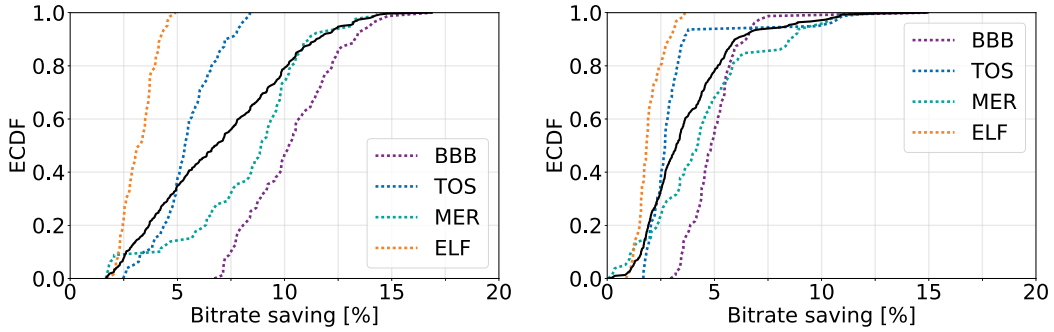
For any video clip, the P-frames contribute the most to the filesize, followed by the B-frames. Although I-frames are the most costly types of frames, i.e., they require much more bytes than P- and B-frames, their overall contribution is the lowest, which stems from the fact that they are used the rarest. In any case, the relatively low overall contribution of I-frames with *NA* can roughly be halved when using *VAR*.

Analogues, the results for encodings with $CRF = 34$ are depicted in Figure 4.7b. It shows that the overall file size is drastically reduced compared to the encoding with $CRF = 16$. The reduced bitrate, which results in lower visual quality, is achieved by a higher compression rate of B- and P-frames. As the I-frames contain the complete image information and as they are used as references, they are compressed to a lower extent, or even left un-compressed. Consequently, the relative contribution of the I-frames to the overall file size increases when increasing the CRF. But similar to the previous case, the amount of bytes needed for I-frames can roughly be halved by *VAR*, resulting in slight decrease of the overall file size for all videos.

4.3.2.4 Reduction of Encoding Overhead

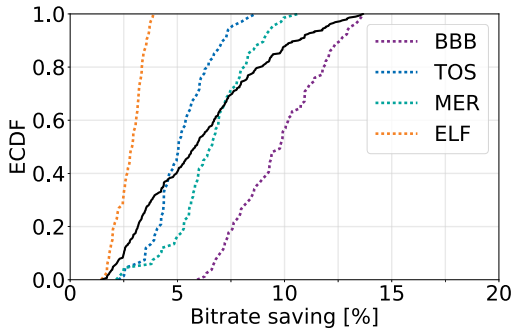
The previous evaluations showed on a small set of examples that I-frames can be saved, leading to a reduction of the overall video file size. In a next step, we evaluate the bitrate reduction on large scale, taking into account the whole data set of encoded videos. For instance, the ECDFs shown in Figure 4.8 comprise the values obtained along all investigated resolutions, (maximum) segment durations, and target bitrates (in the case of CBR) or CRF values (in the case of VBR).

Figure 4.8a illustrates the results for CBR encoded videos, when *VAR* is compared against the fixed approach with the same average segment duration, i.e., *NA*. *VAR* can reduce the average bitrate by up to 16%, which is obtained for the BBB clip. In general, the potential of *VAR* to increase the encoding efficiency is most dominant for BBB. For this video, at least 6% of the bitrate can be economized. The efficiency gain is the lowest for ELF, where the

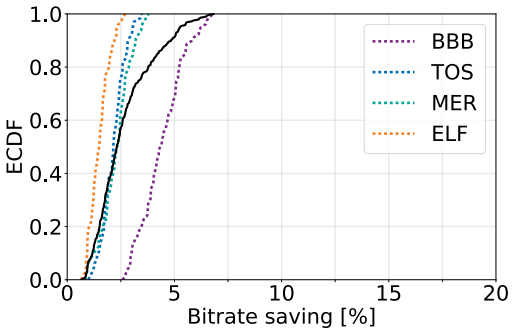


(a) CBR-encoded, compared according to *NA*.

(b) CBR-encoded, compared according to *EM*.



(c) VBR-encoded, compared according to *NA*.



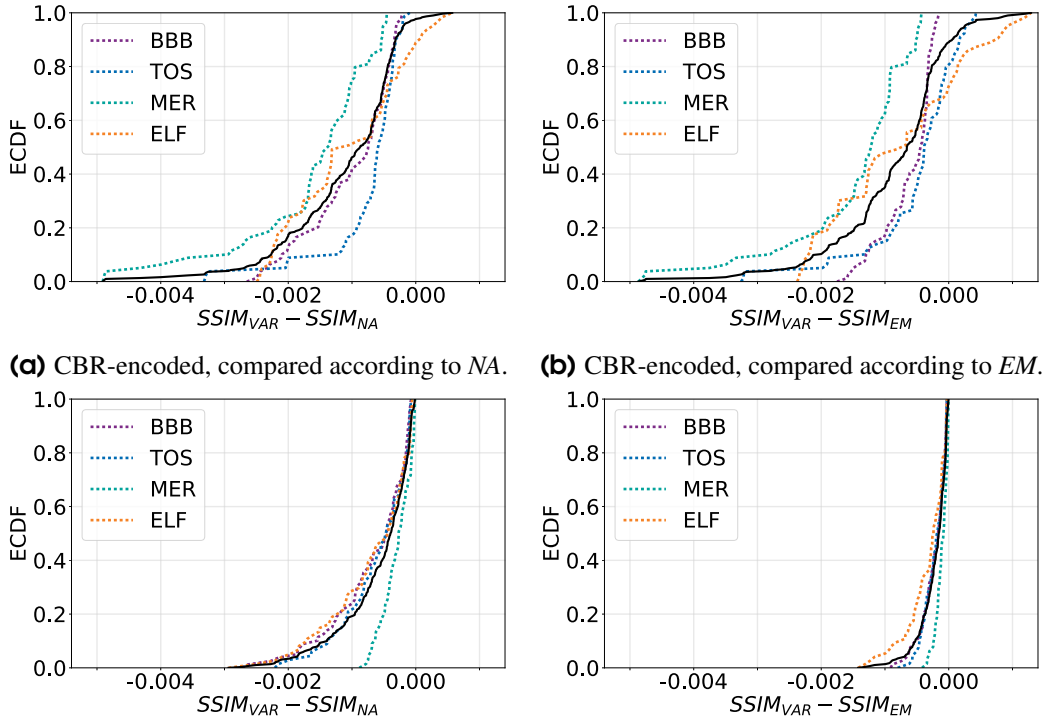
(d) VBR-encoded, compared according to *EM*.

Figure 4.8: Bitrate saving that can be achieved by the variable approach compared to the fixed approach for the different encoding options. Black solid lines denote the overall bitrate saving, colored dotted lines represent the different videos.

bitrate can at most be reduced by 5%. Among all test videos, ELF has the lowest number of I-frames relative to its duration. This results in a lowered potential of *VAR* to reduce the bitrate.

When comparing the variable approach and the fixed approach based on *EM* (Figure 4.8b), the saving in terms of bitrate is lower than in the *NA* case. This is due to the fact that the variable segmentation results in segments that are shorter compared to the respective fixed duration segments. In general, shorter segments imply an increased encoding overhead due to the higher amount of I-frames. Nevertheless, *VAR* yields a median bitrate reduction of about 3% and in the worst case it cannot increase efficiency gain. For instance, *VAR* never results in an increase of the encoding bitrate.

The respective results for the VBR encoding are illustrated in Figures 4.8c and 4.8d. In general, the achieved bitrate reduction when using VBR encoding is lower compared to when using CBR encoding. The maximum for *NA* reduces to 13%, while the maximum for *EM* reduces to roughly 6%. Nevertheless, *VAR* never increases the bitrate requirements. Even if the potential for reducing the bitrate might be very small, as it is the case for ELF with a maximum of about 2.5% according to *EM*, no disadvantaged is induced by using the variable approach.



(a) CBR-encoded, compared according to *NA*.

(b) CBR-encoded, compared according to *EM*.

(c) VBR-encoded, compared according to *NA*.

(d) VBR-encoded, compared according to *EM*.

Figure 4.9: Difference in terms of video quality, expressed as SSIM, for the variable and fixed approach for the different encoding options. Black solid lines denote the overall SSIM loss, colored dotted lines represent the different videos.

4.3.2.5 Impact on Video Quality

In order to fairly compare variable and fixed segment durations, we need to examine whether the I-frame reduction results in a quality degradation for variable segments. Hence, we now analyze the visual quality of fixed- and variably-segmented videos by means of the SSIM metric. This metric compares each frame of a compressed video against the respective uncompressed and distortion-free reference frame. It yields a value ranging between 0 and 1, where 1 means equality to the uncompressed original content, i.e., the highest quality quality. For our evaluations, we calculate the SSIM metric using the respective `ffmpeg` filter, with bicubic upscaling of the encoded video to the 2160p reference.

The plots in Figure 4.9 illustrate the ECDFs of the visual quality loss. Along the x-axis, we denote the absolute SSIM loss, i.e., the magnitude by which the quality is degraded when using *VAR*. For CBR encoded videos, we observe small a quality loss when using *VAR*. The SSIM reduction is in both cases, *NA* (Figure 4.9a) and *EM* (Figure 4.9b), below an absolute value of 0.005. Moreover, for the majority of the analyzed videos, the loss in terms of SSIM is less than 0.001.

For VBR, which specifies a target quality instead of a target bitrate, as in the case of CBR, the quality can better be maintained for the variable approach. Accordingly, the differences

in terms of SSIM are lower. As shown in Figure 4.9c, when comparing *VAR* against *NA*, the SSIM loss is any case below 0.003, and for the majority of the evaluated sequences even clearly below 0.0005. When considering *EM*, the SSIM is degradation is any case less than 0.0015, as shown in Figure 4.9c.

In general, the relationship between SSIM and perceived quality is not linear [72]. For high qualities, i.e., high SSIM values, already small SSIM disturbances may have a high impact on the MOS, while for lower qualities, i.e., low SSIM values, small disturbances are negligible. However, such effects are generally not visible to humans when they are in the order of magnitude which we observe. Hence, the quality degradations incurred by *VAR* are negligible. In the next subsection, where we perform an in-depth investigation of the factors that influence bitrate reduction and quality decrease, we will see that high quality encodings undergo a much smaller SSIM degradation than 0.005.

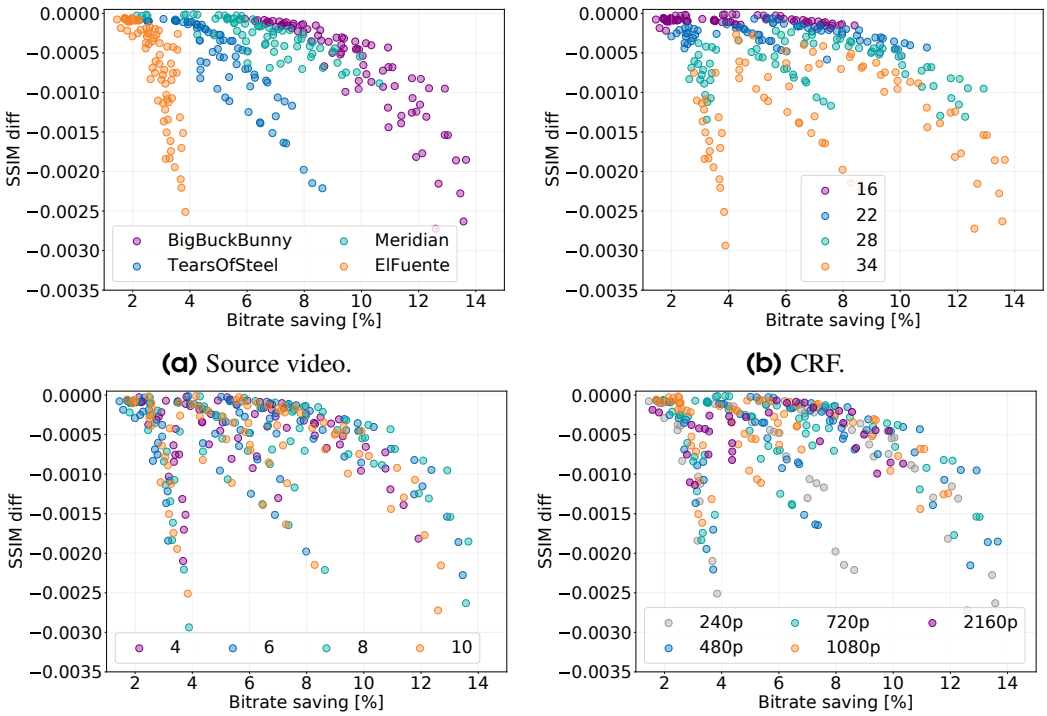
4.3.2.6 Influence Factors on Bitrate and Quality

The evaluations so far show that when using variable segment durations, bitrate can be saved for a slightly lower video quality. In a next step, we evaluate on the example of the VBR encoded videos, in how far the following factors influence the magnitude of bitrate reduction and quality degradation: The source video, the CRF value, the maximum segment segment duration, and finally, the video resolution. The results are illustrated in Figure 4.10, where the y-axis denotes the absolute SSIM loss, and the x-axis represents relative amount of bitrate, which *VAR* can save compared to *NA*.

As already seen above (Figure 4.8), the potential for bitrate saving is highly dependent on the source video, which is in line with the observations from Figure 4.10a. It shows that for ELF, the bitrate reduction is any case below 4%. It hence constitutes the video where the encoding efficiency can be increased the fewest. ELF is followed by TOS and MER, and the BBB brings the highest potential for reducing the bitrate with *VAR*. Further, the plot reveals that for MER, the visual quality remains the most stable with *VAR*.

Figure 4.10b shows a clear influence of the chosen CRF value. With higher CRF values (i.e., lower video quality), variable segment durations tend to degrade SSIM to a greater extent. Nevertheless, this degradation is still too small to be the reason for the significant bitrate reduction we observe. As shown above, the bitrate reduction can be achieved by eliminating I-frames with the more efficient variable method. A second observation from Figure 4.10b is that higher CRF values tend to imply a higher bitrate reduction.

As a third characteristic, we consider the segment duration in Figure 4.10c. Contrary to the two previous factors, i.e., the source video and the CRF value, a clear pattern is now missing. This means that the chosen maximum duration for variable video segments has no strong effect on the bitrate that can be saved compared to the *NA* fixed segmentation. However, there is a slight trend of lower quality degradation if the variable segments do no exceed a duration of 4 seconds.



(a) Source video.

(b) CRF.

(c) Segment duration.

(d) Video resolution.

Figure 4.10: Impact of *VAR* on bitrate and video quality for VBR-encoded videos depending on different video- and encoding-specific characteristics.

Finally, Figure 4.10d shows that the video resolution has no clear visible influence on the bitrate saving achieved with *VAR*, as the plotted data points again do not form a distinct pattern. Whereas a slight trend towards a higher loss of SSIM at lower resolutions can be observed. To summarize, while CRF and the source video itself highly impact the performance of variable segments in terms of bitrate and quality, the effects of maximum durations and video resolution are rather small.

4.4 Impact on Video Streaming Performance

The evaluations in the previous section show that the encoding overhead can be reduced with *VAR*. However, the introduced variability in terms of segment durations results consequently in an enlarged variability of the segments' sizes, which can negatively affect the video streaming performance. As it is not clear whether the reduced bitrate can compensate this enlarged variability, this section evaluates the impact of *VAR* on the video streaming QoE. Via testbed measurements, we generate a large set of video sessions using various network settings and compare the resulting QoE according to the P.1203 model.

4.4.1 Evaluation Methodology

In the following, we describe the methodology for the video streaming experiments. First of all, we present the set of videos chosen for comparing the variable segmentation approach against the fixed approach. Afterwards, we introduce our virtual testbed, along with the video player settings and the used ABR strategies. Finally, we describe the network configurations used throughout the experiments.

4.4.1.1 Video Sequences

During the video streaming evaluations, we confine on comparing the variable approach against the fixed one based on *NA*, i.e., the number of downloaded segments during a video session is practically equal. Furthermore, we use the videos resulting from the CBR encoding, as this is a more applicable encoding method for video streaming, since VBR encoding results in an overall higher bitrate variability.

As previously shown, the video itself has a strong influence on the performance of variable segment durations during the encoding process (cf. Figure 4.10a). To account for this influence factor during the video streaming process, we consider all of the four source videos for our analysis. For each video, we use the variable video representation with the highest maximum segment duration, i.e., 10 seconds. Note that in terms of encoding efficiency, the effect of the maximum duration is negligible (cf. Figure 4.10c) compared to the effect of the source video or target quality. However, the larger the maximum duration, the higher is the variability of the resulting segments' sizes, which negatively affects video streaming performance [96]. Hence, the evaluations using the variable videos with a maximum duration of 10 seconds can be seen as a "worst case scenario" with respect to the variability of the segments' durations and sizes. The coefficient of variation of the segment sizes is larger with *VAR* for all of the selected video clips. For the BBB clip, it increases from 0.43 (*NA*) to 0.74 (*VAR*), and for TOS from 0.45 to 0.76, respectively. In the case of MER, the coefficient of variation with *NA* is 0.69 and 0.77 with *VAR*. Finally, *VAR* increases the segment size variability for ELF from 0.52 to 0.55, being the lowest increase among all clips used during the video streaming measurements.

To determine the bitrate ladder (i.e., the resolution-bitrate pairs selected for video streaming), we utilize the selection method presented in [128]. For each video, we choose 5 different quality levels, i.e., bitrate/resolution combinations, which we obtain from the resulting bitrate ladder. Table 4.4 illustrates the resolutions and bitrates used for the different quality levels for the *VAR* videos. Please note that the quality levels for the *NA* videos only differ in the sense that the bitrates are slightly higher on each level, due to the higher encoding overhead. The resolutions are equivalent between *VAR* and *NA* along the different quality representations. We omit video representations with a resolution of 2160p, as this resolution is not supported by the P.1203 standard.[11]

[11] Current developments in the ITU-T P.1204 recommendation series will address 4K/UHD video.

Table 4.4: Bitrates and resolutions selected for the streaming measurements.

	BBB		TOS		MER		ELF	
L	*Res*	*BR*	*Res*	*BR*	*Res*	*BR*	*Res*	*Br*
0	480p	215 kbps	240p	234 kbps	720p	164 kbps	240p	291 kbps
1	720p	406 kbps	480p	354 kbps	720p	342 kbps	480p	403 kbps
2	1080p	797 kbps	720p	689 kbps	1080p	492 kbps	720p	942 kbps
3	1080p	1.6 Mbps	720p	1.4 Mbps	1080p	2.0 Mbps	720p	2.1 Mpbs
4	1080p	3.4 Mbps	1080p	2.7 Mbps	1080p	12.6 Mbps	1080p	3.5 Mbps

4.4.1.2 Measurement Environment and Video Player Settings

We run our measurements in a virtual testbed, consisting of three VMs, set up via *Vagrant* and *VirtualBox*. One of VMs acts as the server hosting the videos, one as the HAS streaming client, and the third VM acts as a network emulator. The latter connects client and server and allows to emulate different network settings, i.e., rate limiting using the Linux traffic control[12]. The client runs the browser-based DASH reference player `dash.js`[13] in version v3.0.0. We modified the player so as to log all relevant metrics for QoE computation, such as playback quality or video stallings. For the sake of scalability, and to allow streaming tests without actually playing back the video (e.g., when running on a server where no display is attached), the browser runs in headless mode. To allow the client to request videos in headless mode, we use *Puppeteer*[14], which runs on top of *Node.js*.

The `dash.js` player implements the three following ABR strategies: a buffer-based solution according to BOLA [95], a throughput-based, and a hybrid solution.[15] In order to be able to draw more generic conclusions, we run testbed measurements with each of the three available strategies. In following parts of this work, we will refer to them as *BOLA-ABR*, *throughput-ABR*, and *hybrid-ABR*. We set the initial buffer threshold to 12 seconds, and the stable buffer time, i.e., the internal buffer target the player tries to reach, to 30 seconds. The maximum buffer time is set to 45 seconds, i.e., the client will pause segment requests when this threshold is reached.

4.4.1.3 Network Settings

We test the feasibility of variable segment durations for adaptive streaming with fluctuating bandwidth capacities, which allows to capture the behavior in a more stressful manner. We use realistic bandwidth traces [111] which have been collected in a 3G network during commute with different types of public transport. We scale them so as to achieve an average rate of the $\{1,2,4,6\}$-fold of the lowest quality's bitrate of each of the four *VAR* test videos. Analogous to the previous chapter, we refer to these bandwidth limit settings as

[12] https://linux.die.net/man/8/tc
[13] https://github.com/Dash-Industry-Forum/dash.js
[14] `https://github.com/puppeteer/puppeteer`
[15] `https://github.com/Dash-Industry-Forum/dash.js/wiki/ABR-Logic#primary-rules`

bandwidth provisioning factor and denote it as a, i.e., $a \in \{1,2,4,6\}$. Additionally, we limit the available bandwidth to the 1-fold of the lowest quality's bitrate for each *NA* video. The very low bandwidth settings allow a comparison of *NA* and *VAR* in those scenarios, where hardly any other than the lowest quality can be downloaded and where the ABR's behavior is negligible for the streaming performance. We confine on $a = 6$ as the highest rate, as this bandwidth configuration already triggers the ABR to choose between different levels that yield a decent video quality. Hence, these scenarios allow to study the impact of *VAR* on the heuristic's behavior and consequently on the quality adaptation and resulting video streaming performance.

From the trace data set, we choose three replicas, i.e., same commute path and vehicle but measured on a different day, of the traces *car*, *ferry*, and *tram*. We furthermore define three different starting points for each of the traces, namely from the beginning, i.e., second 0, and two randomly chosen start points. The traces are looped, i.e., if the end of the trace is reached, it starts again from the beginning. For each trace replica and each start pointing, three measurement runs are performed. Hence, having a single different trace type (car, ferry, or tram) which is applied with three different starting points, this results in 27 measurement runs for a single configuration. A specific configuration is a combination of the bandwidth provisioning factor a, the video along with its segmentation option (*VAR* and *NA*), the applied adaptation strategy, and finally the trace type. This results in a data set comprising more than 7,000 streaming sessions with a duration of at least 8 minutes each.

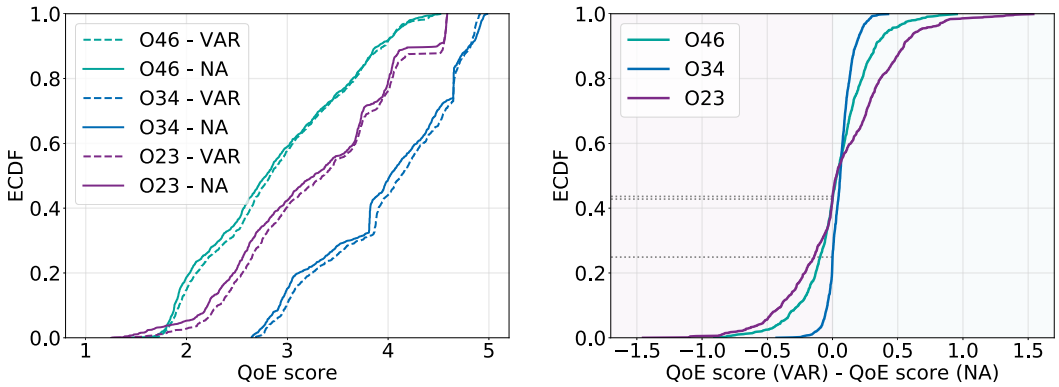
4.4.2 Evaluation Results

In the following, we compare the performance of variable and fixed segment durations with respect to the obtained QoE obtained. Thereby, we evaluate the QoE of each streaming session according to the ITU-P.1203 model in mode 3, which takes video frame-level characteristics, such as the frame types, frame sizes, and the quantization parameter (QP) values on a per-macroblock scale into account. As we omit the audio track for the videos, the QoE model per default assumes a constant high audio quality when computing the audio-visual quality score (O34).

Our in-depth analysis of all three ABRs available in the dash.js reference implementation, i.e., *hybrid-ABR*, *BOLA-ABR*, and *throughput-ABR*, showed that there are only slight differences in terms of how *VAR* performs compared to *NA*. Table 4.5 denotes the median improvements for the overall QoE score, as well as for the two diagnostics scores, achieved by *VAR*, i.e., *QoE(VAR)-QoE(NA)* for all test runs using the different ABR strategies. We furthermore indicate the confidence intervals on a 95% confidence level. The medians of the different QoE scores differ only slightly and the corresponding confidence intervals overlap in most of the cases. Particularly for the overall QoE O46, the confidence intervals for all of the ABRs overlap, showing that the impact of the ABR is not significant. For that reason, we confine the following detailed streaming analysis on *hybrid-ABR*, which is the default configuration of dash.js

Table 4.5: Median QoE improvements (QoE(*VAR*) - QoE(*NA*)) and confidence intervals (CI) for the different QoE metrics over all runs.

Score	Description	Improvement ± CI		
		hybrid-ABR	*BOLA-ABR*	*throughput-ABR*
O23	Stalling quality	0.034 ± 0.010	0.011 ± 0.011	0.008 ± 0.008
O34	Audio-visual quality score	0.048 ± 0.005	0.043 ± 0.003	0.046 ± 0.005
O46	Overall quality score	0.035 ± 0.012	0.015 ± 0.015	0.011 ± 0.013



(a) Absolute QoE scores.

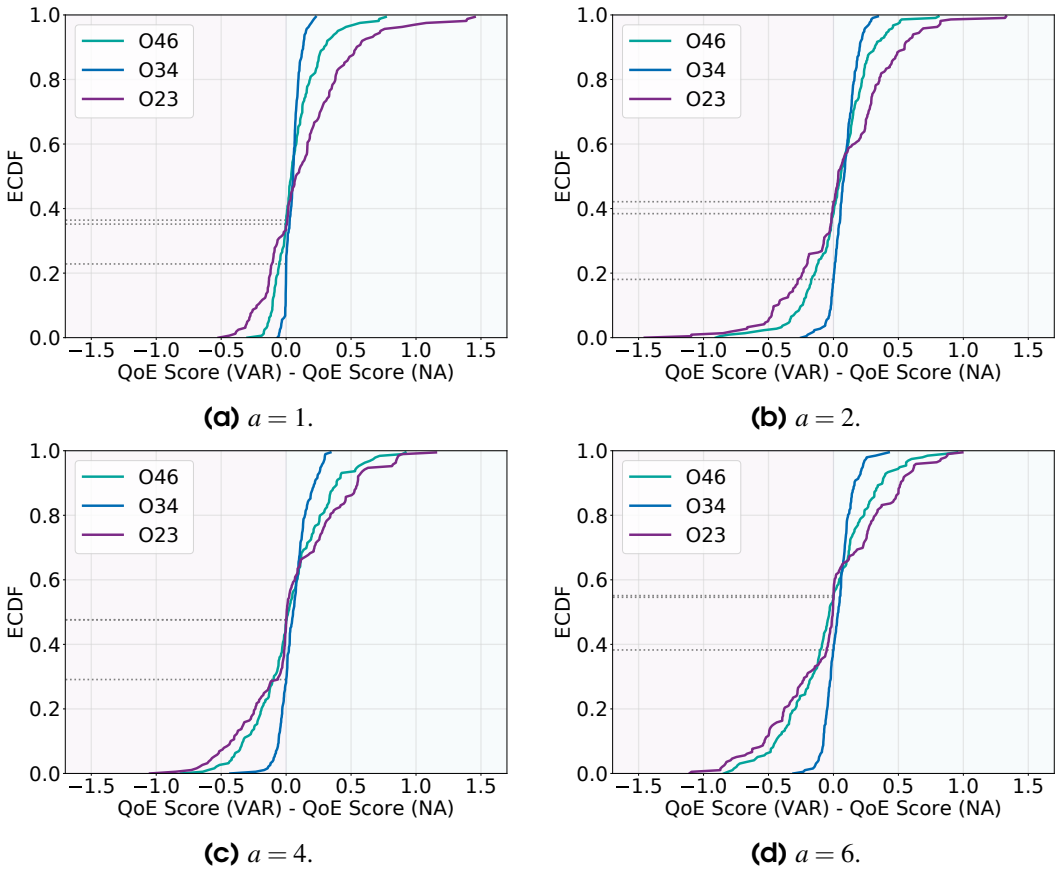
(b) Difference of QoE scores.

Figure 4.11: Absolute value and differences of the QoE scores obtained from the measurements using the adaptation strategy *hybrid-ABR*.

First, we comprehensively analyze the QoE scores by considering all performed measurement runs. In a next step, we investigate the impact of *VAR* under different settings of the available bandwidth. Finally, we discuss in more detail the results obtained for one specific scenario.

4.4.2.1 Overall Analysis of QoE Scores

Figure 4.11a illustrates the different QoE scores obtained for all measurement runs with variable segment durations (*VAR*) and the respective fixed segment durations using *NA* comparison and the *hybrid-ABR* logic. The x-axis represents the values on MOS scale, the y-axis denotes the ECDF. The dotted lines, representing the values for *VAR*, are slightly shifted towards the right along the x-axis compared to the respective solid lines. Hence, *VAR* tends to yield higher MOS values for the three different QoE scores. While the median value for O46, i.e., the overall quality, is 2.806 for *NA*, this value can slightly be increased to 2.853 by *VAR*. In terms of stalling quality, denoted as O23, *NA* achieves a median value of 3.346, which also can slightly be improved by *VAR*, which achieves a median value of 3.404. More significant improvements using the variable approach can be seen for O34, i.e., the audio-visual quality score. While the median for *NA* is 4.0, this value increases to 4.12 for *VAR*.



(a) $a = 1$. **(b)** $a = 2$. **(c)** $a = 4$. **(d)** $a = 6$.

Figure 4.12: Differences in terms of the QoE scores for different settings of the bandwidth provisioning factors a.

Figure 4.11b illustrates the absolute differences of the QoE scores obtained for all measurement runs. Negative values along the x-axis denote an impairment induced by *VAR*, positive values indicate an improvement of the QoE. The most significant absolute differences between *VAR* and *NA* can be observed for the stalling quality (O23). For this score, the absolute deviations can be up to 1.5 on MOS scale and an improvement by *VAR* can be achieved for 56% of the investigated streaming sessions. The audio-visual quality score, i.e., O34, can be improved by *VAR* in 75% of the tested cases, however, the magnitude of impairment or improvement is in any case below 0.5 and hence, significantly lower compared to O23. Finally, the overall quality score O46 can be improved for 57% of the sessions.

Overall, the median improvement achieved by *VAR* with *hybrid-ABR* is 0.034 ± 0.01 for O23 and 0.048 ± 0.005 for O34. The median improvement of the overall QoE score, i.e., O46, is 0.035 ± 0.012 (cf. Table 4.5). As none of the denoted confidence intervals includes 0, we can conclude that the improvements achieved by *VAR* are significant for all considered QoE metrics.

4.4.2.2 QoE Scores Obtained with Different Bandwidth Capacity Limits

In order to better understand the impact of *VAR* in the different scenarios, we evaluate the obtained QoE scores for different bandwidth capacities. Figure 4.12 shows the difference $QoE(VAR) - QoE(NA)$ for different rate limits, i.e., settings of a, while Table 4.6 denotes the average values of the QoE scores obtained with *VAR* and *NA*.

Figure 4.12a shows that for a bandwidth provisioning factor of $a = 1$, the overall QoE score (O46) can be improved in 64% of the cases. The maximum improvement of O46 that can be observed for this rate is 0.767, while in the worst case, *VAR* yields an impairment of the QoE by 0.297. The differences between *VAR* and *NA* in terms of O34, i.e., the audio-visual quality score, are relatively small. This is due to the fact that the small bandwidth capacity hardly leaves room for streaming on any other than the lowest quality level. Nevertheless, for the majority of the streaming sessions *VAR* improves the O34 scores and additionally, the magnitude of improvements is higher than the magnitude of impairments. For the O23 score, *VAR* yields an improvement in more than 60% of the streaming sessions. While it can be increased by up to 1.452, it is never worsened by more than 0.52. This indicates a significantly lower number of video interruptions due to the increased encoding efficiency and the consequently reduced bandwidth requirements.

Figure 4.12b shows the results for a bandwidth provisioning factor of $a = 2$. The visual quality (O34) can now be improved for 82% of the measurement runs. This indicates that the video buffer can be filled up more quickly, allowing the client to choose higher qualities more often in the case of *VAR* compared to *NA*. Besides, we observe an increase of the stalling quality (O23) for 58% of the test runs. Overall, this results in an improvement of O46 in 62% of the cases.

For an average rate corresponding to $a = 4$, as illustrated in Figure 4.12c, both, the overall quality score (O46) and the stalling quality score (O23), are improved by *VAR* for 52% of the test runs. The audio-visual quality can be improved in roughly 70% of the investigated cases. Hence, compared to the two previous cases, i.e., $a = 1$ and $a = 2$, the potential of *VAR* to improve QoE is significantly lower. This indicates that the achievable improvement is decreasing as the bandwidth capacity increases.

This assumption can be confirmed when evaluating the streaming sessions obtained with the highest bandwidth availability, i.e., when $a = 6$, as illustrated in Figure 4.12d. We now observe the first case, where the fixed approach outperforms the variable approach for the majority of the test runs. In 55% of the cases, *NA* yields a higher overall quality score (O46) than *VAR*. Furthermore, the stalling quality (O23) is in 55% of the cases higher with *NA*, than with *VAR*. However, in 62% of our scenarios, *VAR* still increases the audio-visual quality score.

The overall QoE (O46) is mainly affected by the stallings [84]. This is also noticeable in our evaluations, as O46 and O23 have a similar behavior for the shown cases in Figure 4.11b and Figure 4.12. If O23 can be increased by *VAR* for a certain share, the share with which

Table 4.6: Average values for the different QoE scores obtained with *hybrid-ABR*. Bold numbers represent the respective higher value.

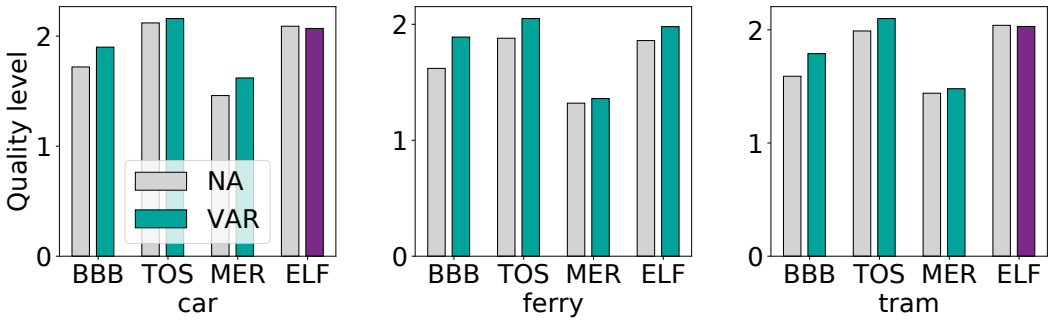
	O23		**O34**		**O46**	
a	*VAR*	*NA*	*VAR*	*NA*	*VAR*	*NA*
1.0	**2.510**	2.370	**3.582**	3.530	**2.178**	2.105
2.0	**3.585**	3.518	**3.963**	3.882	**2.973**	2.931
4.0	**3.873**	3.826	**4.370**	4.309	**3.435**	3.399
6.0	3.923	**3.939**	**4.533**	4.496	3.587	**3.618**

O46 can be increased is similar to that. The degradation of O46 in scenarios with increasing available bandwidth might be due to an increase of stallings, resulting from a too optimistic quality adaptation of the heuristic with *VAR* videos. The resulting higher visual quality, however, cannot compensate the increased number of stallings and hence, the overall QoE is decreased with *VAR* compared to the fixed approach.

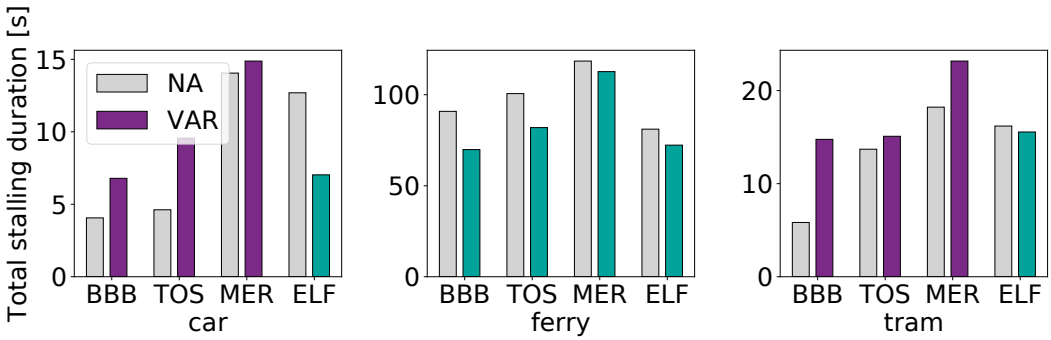
Table 4.6 summarizes the average QoE obtained with *VAR* and *NA* for different settings of the bandwidth provisioning factor a. As long as $a \leq 4$, *VAR* yields higher values for all three types of types of scores. However, when $a = 6$, the average stalling quality is decreased by *VAR*. Simultaneously, this is the only scenario where the overall quality score is impaired with the variable approach. Although *VAR* can still improve the audio visual score, this visual quality improvement cannot compensate for the induced stallings. However, to support this hypothesis, we deeper investigate the scenario with $a = 6$ in the following.

4.4.2.3 Detailed Investigation of High Bandwidth Capacity Scenarios

Figure 4.13a denotes the average quality level obtained for the different source videos with different trace types. In any case, *VAR* yields a higher visual streaming quality on average compared to *NA*, except for two cases. The first one is the ELF clip in conjunction with the *car* trace, the second case is again the clip ELF, but for the *tram* trace. Next, we focus on the total stalling duration, which is denoted in Figure 4.13b. For the trace *ferry*, the variable approach is capable to reduce the overall re-buffer time, independent of the underlying source video. However, for the traces *car* and *tram*, *VAR* leads to an increase of the total stalling duration for the clips BBB, TOS, and MER. Only for ELF, the streaming performance in terms of stallings can be improved by *VAR*. Please note that ELF is simultaneously the only video, in which *VAR* did not increase, but very slightly decrease the average video quality level. This supports our claim from above, that there tends to be a too optimistic quality selection with *VAR*, leading to a higher re-buffer time, and hence to a reduction of the QoE in general. Please not in this context that a stalling duration increase from 4s to 6s (e.g. BBB with the car trace in Figure 4.13b), can have a drastic negative negative impact on the overall QoE. In the following, we further investigate the assumption of a too optimistic quality selection, which results into stallings, by taking a closer look on the average video buffer filling levels.



(a) Average quality level.



(b) Total stalling duration.

Figure 4.13: Quality level and stalling duration for $a = 6$. Gray bars show the results for *NA*. Green bars denote an improvement by *VAR* compared to *NA*, purple bars denote an impairment.

The *hybrid-ABR* selects the next segment's quality based on both, the throughput measured during the last segment downloads well as the current buffer level. As we configured the target buffer level as 30 seconds, quality switches are likely to happen shortly before this level is reached. Figure 4.14 illustrates the probability of a buffer level above 26 seconds. For the three clips BBB, TOS, and MER, the probability of having a buffer level near the target buffer is higher for *VAR* than for *NA*. In contrast, for ELF, the probability of buffer values higher than 26s is similar for *VAR* and *NA*, but slightly lower with *VAR*. This strengthens our hypothesis that with *VAR*, the overall QoE score (O46) decreases for higher rates due to a too aggressive quality adaptation. As the average bitrate for *VAR* is lower compared to *NA*, higher buffer values can be reached faster. This triggers the heuristic to switch to a higher quality, which can lead to stalling, especially in environments with varying bandwidth capacity and if the downloaded segment is of comparably long duration. Please note the increased variability in terms of segment sizes for the *VAR* approach. As stated above, the coefficient of variation for the segments sizes is increased by 0.31, for BBB and TOS and by 0.08 for MER. With higher quality levels, and hence higher bitrates, the absolute variation of the segment sizes increases. For the ELF clip, i.e., that clip in which *VAR* also improves the QoE for higher bandwidth capacities, the segment sizes' coefficient of variation is only decreased by 0.02. To summarize, the combination of a fast buffer ramp-up with *VAR* and the potentially highly differing segment sizes at higher quality levels, results in a

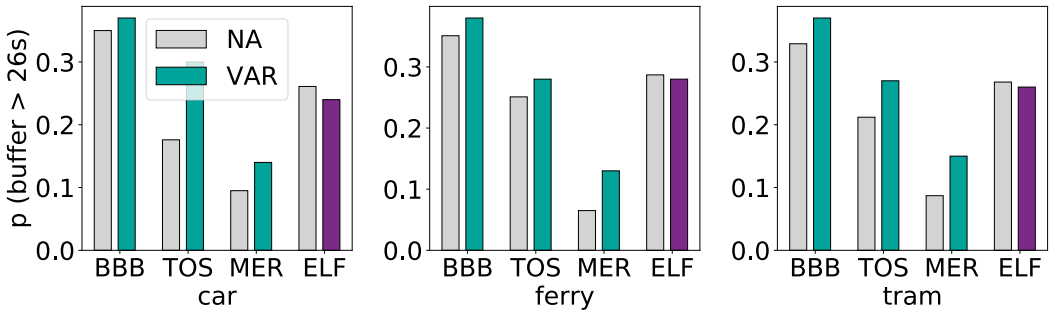


Figure 4.14: Probability for buffer levels nearby the target buffer. Gray bars represent the probability for *NA*. Green bars denote a higher probability with *VAR*, purple bars a lower probability with *VAR* compared to *NA*.

worsened streaming performance compared to *NA*. This can be overcome by adjusting the buffer thresholds for quality switches to the expected segment size variation. For instance, the higher the variability, the more conservative the client should switch to a higher quality level.

4.5 Lessons Learned

In this chapter, we performed a holistic analysis of the potential benefits of using a content-depending video segmentation technique, which results segments of variable durations. The conducted study is comprised of two parts: the first one focuses on the video encoding and segmentation process, while the second one addresses the implications of variable segment durations on the HAS performance during streaming.

In the scope of the first part, we showed that the variable approach is capable of significantly reducing the number of costly I-frames during encoding, which consequently results in a reduced file size as compared to the state-of-the-art mechanisms relying on segments of equal durations. In order to draw generalizable conclusions in terms of the achieved reduction of the bitrate requirements, we encoded a set of representative video clips using a vast number of encoding- and segmentation-specific parameter combinations. The analysis of the resulting data set, comprising nearly 2,000 encoded video sequences, revealed that the bitrate can be reduced by up to 15% by the variable approach. Furthermore, we identified the chosen CRF value and the underlying source video as the main influence factors for the encoding efficiency gain, while there is only a negligible impact of the specified maximum segment duration and the video's resolution. Although the improvements achieved by the variable approach are minor in certain cases, the average bitrate was never increased for any of the evaluated video sequences. Hence, deploying such a solution is in any case beneficial. Even a marginal relative bitrate reduction of a few percent, results in total in a significant absolute cost reduction for storing and transmitting videos, given the huge amount of video content provided online and the massive share of HAS on the overall IP traffic.

In the second part of our study, we investigated the impact of variable segment duration on the HAS performance. By means of testbed measurements, we created a comprehensive set of more than 7,000 streaming session, including different videos, three diverse ABR strategies, as well as various bandwidth characteristics. We learned that the approach relying on variable segment durations can clearly outperform the conventional approach in scenarios with low bandwidth capacities. There, the overall QoE score could be improved for 64% of the streaming sessions, with an absolute gain of up to 0.767 on the five point MOS scale, while the impairments occurring in the remaining 36% of the evaluated sessions never degraded the QoE by more than 0.297. This achievement is mainly due to the lower number of stallings with the variable approach, resulting from the reduced bitrate. However, we found that the variability of the segments' sizes increases with the content-dependent technique. This effect becomes stronger with visually better quality levels, which are encoded with higher bitrates, and hampers the ABR strategy to carry out optimal decisions. Our analysis showed that in scenarios with high bandwidth capacity, the ABR mechanism acted too optimistic in terms of switching to a higher quality level. This increases the risk of a stalling when the requested segmented has a comparably long duration and thus, takes longer to be downloaded due to its larger size. However, this issue is not a drawback of variable segment durations per se, as it can be addressed by a proper tuning of the quality switching thresholds or by deploying an ABR strategy which is capable of considering the segment sizes and their variability, such as the mechanism proposed in [65]. Hence, we conclude that the HAS ecosystem can further be optimized by considering the underlying video during the segmentation process, as this allows to reduce the bitrate requirements and – given the same network conditions – to deliver a better QoE as compared to the content-agnostic state-of-the-art approach.

5

Machine Learning for QoE Estimation in 5G Networks

The two previous chapters presented approaches related to QoE assessment and QoE optimization. More specifically, the analytical model proposed in Chapter 3 allows for understanding the interplay of the HAS parameters and their impact on QoE and can thus be used by an AP to derive an optimized parameter setting. The content-depended segmentation approach discussed in Chapter 4 can be exploited by an AP to deliver a better QoE due to the reduced bitrate requirements. Besides the AP, the NP is an important stakeholder in the Internet ecosystem, which is driven by business incentives to deliver a good QoE to its customers. However, while the AP has capabilities to monitor QoE-relevant application-level metrics, such as video interruption times, and to collect user ratings, the NP only has access to network-level data. Although models exist, which allow a mapping from collected QoS metrics to QoE, they are typically only available for a limited number of services, are cumbersome to design and update, and focus only on key network features or aggregated monitoring information when deriving the QoE. They allow for identifying service degradation, but their general applicability for root cause analysis or as a basis for fine-grained resource control is limited. As a consequence, these traditional models, for example, cannot be used for self-driven networks, which implement automated control-loops for triggering QoE-aware network-control actions based on the derived QoE.

Evolving networking technologies and architectures, like the 5G system (5GS), introduce new NFs, providing improved monitoring capabilities and novel ways for application-network

interactions. NFs potentially enabling an improved QoE monitoring are the Application Function (AF) and the Network Data Analytics Function (NWDAF). The AF implements a standardized interface which enables the interaction between the 5GS and third parties, such as APs. For instance, it can be used by APs to communicate any QoE-relevant information to the MNO. The NWDAF is capable of collecting fine-grained and real-time network statistics from other 5G control- and user-plane NFs. Besides, it can perform complex analytics upon the gathered data. While using the AF to receive detailed application information on a per-device basis is theoretically possible, such an exchange of data does not scale and would easily overload the AF or the mobile network control plane. Moreover, it is to be expected that an AP is not be willing to constantly expose the QoE or relevant KPIs, which indicate the performance of its application. Hence, it is of upmost importance to rely on network monitoring data available at the NWDAF and to enrich this data so to allow an accurate and fine-grained QoE estimation for a diverse set of applications.

The relationship between the different network KPIs and QoE is complex and the significance of a measured parameter (or a combination of the measured parameters) for the resulting QoE is highly dependent on the used service, the user context, the value of the measured parameter, and the QoE score itself. Using Machine Learning (ML) is a promising approach to efficiently learn the complex interactions between the available network statistics and the QoE in different contexts and is additionally capable of overcoming the limitations of traditional QoE models, which derive the QoE purely based on a pre-defined mapping of fixed inputs. Once trained and deployed, the ML model allows the MNO for estimating the service-specific QoE for any user from its collected statistics. Although the 5G architecture provides the infrastructure for such an ML-driven approach, it is unclear how reliable and efficient such a QoE estimation can be performed for diverse services with their individual network requirements.

In this chapter, we shed light on the challenges and prospects for introducing an ML-based QoE estimation in 5GS. We generally investigate if an ML-driven QoE estimation can realistically be performed based on the information accessible to an MNO via the NWDAF. We first identify the challenges and design criteria linked to the ML deployment from an operator's perspective. Secondly, we identify how good these challenges can be met by a representative set of five different regression techniques, ranging from simple linear models to highly complex neural networks. For instance, we study how accurate and how efficient these regression techniques can estimate the QoE. Thereby, our evaluations are based on data generated from simulation activities, taking into account the heterogeneity resulting from different user characteristics and running services.

The rest of this chapter is structured as follows. Section 5.1 introduces the ML models used throughout the evaluations and presents related work. Next, we elaborate in Section 5.2 on the challenges and design criteria from an MNO's point of view and describe how they are addressed in the scope of this chapter. Section 5.3 introduces the methodology and Section 5.4 describes our ground-truth data set. Subsequently, we present the results of our studies in Section 5.5. The chapter concludes with the lessons learned in Section 5.6.

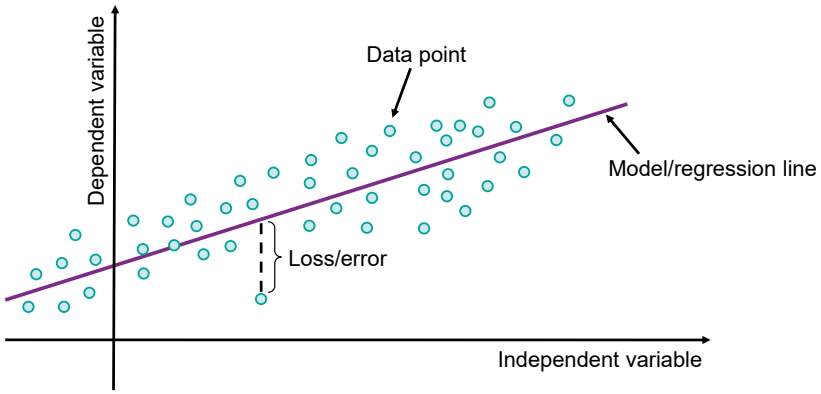


Figure 5.1: Schematic illustration of a simple regression model. It expresses the relationship between the dependent and the independent variable by means of the regression line which minimizes the overall loss.

5.1 Background and Related Work

In the following, we briefly describe the basic working principle behind the regression technique used throughout the evaluations of this chapter. Next, we present the performance metrics used to assess the accuracy obtained with the different ML model and subsequently discuss related works.

5.1.1 Regression Techniques

Regression analysis is a widely used concept in the field of Machine Learning. Regression models learn the relationship between a dependent variable and one ore more independent variables, which are often referred to as predictors, co-variates, or - as throughout the remainder of this chapter - *features*. As the dependent variable, i.e., the outcome, is part of the training process, regression techniques fall in the field of supervised learning. Having trained a regression model then allows to predict the unknown response from features at deployment. Features can be raw observations or their predefined transformation. Generally spoken, regression techniques find a mathematical equation that expresses the relationship between the available features and the outcome. A schematic illustration is given in Figure 5.1. It depicts a single feature (independent variable) on the x-axis and the outcome (dependent variable) on the y-axis. The regression line indicates their relationship, which has been learned by the regression model in such a way, that a specific loss function, taking into account the distance between the regression line and each of the data points, is minimized. In order to prevent over-fitting of models, they typically apply *regularization*, which puts an additional penalty to the loss function so to decrease the sensitivity towards the input features [129]. An unknown outcome is then predicted solely based on the given feature value (independent variable) by simply applying the learned function to it.

In the course of this chapter, we use five different regression techniques, which are briefly introduced in the following. For two of them, the response is expressed as a linear function of the features. Three of the models are non-linear, which allows them to learn more complex relationships.

5.1.1.1 Least Absolute Shrinkage and Selection Operator

LASSO [130] is a linear regression model trained with the L1 regularizer, the sum of absolute values of the weights (or the regression coefficients), in addition to the mean squared error (MSE) of the prediction. The L1 regularizer is known to induce *sparsity*, meaning that many of the learned weights are zero, and therefore the corresponding features are completely neglected when estimating the response, i.e., the QoE in our case. The number of zero weights can be tuned by the regularization parameter λ, which controls the strength of shrinkage and sparsity. By doing so, LASSO does not only help to reduce over-fitting, but can also be used for feature selection.

If the goal is to estimate the QoE y from a limited number of features as accurate as possible under a limited total cost of making the necessary features available (in the run time), this can be expressed in a standard machine learning formulation, called the regularized linear regression. Let c_d be the computation/communication cost of the d-th feature x_d. Then, we want to solve the following problem:

$$\begin{aligned} &\min_w L \equiv |y - \textstyle\sum_{d=1}^{D} w_d x_d|^2 \\ &\text{subject to} \textstyle\sum_{d=1}^{D} c_d \delta(w_d \neq 0) \leq \lambda_0. \end{aligned} \tag{5.1}$$

Here, $w = (w_1, \ldots, w_D) \in \mathbb{R}^D$ is the regression parameter, $\lambda_0 > 0$ is the total cost budget, and $\delta(\cdot)$ is the indicator function giving one if the event is true and zero otherwise. Unfortunately, the problem (5.1) is known to be intractable (NP-hard) because the non-convex constraint, for which a convex surrogate is often used.

$$\begin{aligned} &\min_w L = |y - \textstyle\sum_{d=1}^{D} w_d x_d|^2 \\ &\text{subject to} \textstyle\sum_{d=1}^{D} c_d |w_d| \leq \lambda_1, \end{aligned} \tag{5.2}$$

where y and $x = (x_1, \ldots, x_D)$ are assumed to be standardized, so that the means and the standard deviations of y and $\{x_d; d = 1, \ldots, D\}$ are all zero and unity, respectively. The problem (5.2) is now convex, and can be simplified as

$$\min_w L \equiv |y - \textstyle\sum_{d=1}^{D} w_d x_d|^2 + \lambda \|w\|_1, \tag{5.3}$$

for the regularization parameter $\lambda > 0$ (which should be appropriately set so that it matches λ_1) and the pre-processed features such that the means and the standard deviations of $\{x_d\}$

are zero and c_d^{-1}, respectively. The second part of the sum in 5.3 is called the ℓ_1-norm

$$\|w\|_1 \equiv \sum_{d=1}^{D} |w_d| \equiv |w_1| + |w_2| + |w_3| + ... + |w_D|$$

and is often used in machine learning in order to induce sparsity—many of $\{w_d\}$ get exactly zero as λ grows. It was proved that the solution of the problem (5.3) is piece-wise linear to λ, and various efficient solvers have been developed and available online.

The number $\sum_{d=1}^{D} \delta(w_d \neq 0)$ of used features is monotonically decreasing with respect to λ, providing the best feature set for each number of used features. When the total cost budget λ_0 is given, it allows for evaluating the total cost $\sum_{d=1}^{D} c_d \delta(w_d \neq 0)$ for each of those best feature sets, and for choosing the one giving the lowest regression error under the budget. Thus, LASSO can be used to find a feature set which is as cheap as possible, allowing to tune the cost versus accuracy trade-off for estimating the QoE.

5.1.1.2 Linear Ridge Regression

LRR [131] is, same as LASSO, a linear regression model. However, while LASSO relies on the L1 regularizer, LRR uses the L2 regularizer. Hence, while the problem which is solved by LASSO as denoted in 5.3, the according problem solved with LRR is defined as follows:

$$\min_w L \equiv |y - \textstyle\sum_{d=1}^{D} w_d x_d|^2 + \lambda \|w\|_2^2. \tag{5.4}$$

The second part of the sum in 5.4 is the regularization applied by LRR, which is the square of the ℓ_2-norm, which is defined as follows:

$$\|w\|_2 \equiv \sqrt{\sum_{d=1}^{D} {w_d}^2} \equiv \sqrt{{w_1}^2 + {w_2}^2 + {w_3}^2 + ... + {w_D}^2}.$$

For instance, while the ℓ_1-norm, which is used with LASSO, is calculated as the sum of absolute values in the vector, the ℓ_2-norm is calculated as the square root of the sum of the squared vector values. The main difference between them is that L1 regularization estimates the median of the data, while L2 regularization estimates the mean of the data for over-fitting prevention and that L2 hardly shrinks any weight to zero. It cares more about driving big weights to small values, and tends to give small but well distributed weights. By doing so, LRR tends to provide better prediction accuracy than LASSO, while it cannot be directly used for feature selection, as non of the features are fully eliminated. If no penalty is given, i.e., $\lambda = 0$, LRR performs ordinary least squares, while increasing values of λ decrease the sensitivity of the model to the input features. That is, if λ is set very large, the trained model will tend to under-fit, making it a crucial task to find an appropriate setting for that parameter.

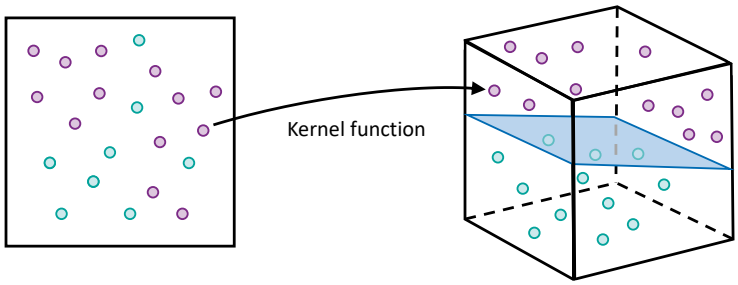


Figure 5.2: Schematic illustration of separable data points after transforming them by means of a kernel function.

5.1.1.3 Kernel Ridge Regression

KRR [131] is a kernel regression model, which is - similar as LRR - trained with the L2 regularizer. Sometimes, data is not separable in an n-dimensional space, but in higher dimensions, as denoted in Figure 5.2. For that reason, KRR applies the kernel trick, which is the dot product of the transformed vectors in the higher dimensional space. That is, with the kernel, the transformation of the data points into the feature space is only performed implicitly. The kernel trick allows for operating in the original feature space without computing the high dimensional mapping, offering a more efficient and less expensive way to non-linearly transform features into high dimensional space. The kernel regression is equivalent to linear regression applied to a high, possibly infinite, dimensional space into which the original features are non-linearly mapped. By choosing an appropriate kernel, this model can approximately express any smooth function. Upon others, laplacian, polynomial, exponential, or sigmoid kernel functions are typically used. In our analysis we use the Gaussian kernel, another popular option, with the bandwidth parameter γ, with which KRR is capable of learning any non-linear relationships between the features and the response.

5.1.1.4 Support Vector Regression

SVR [132] is a similar model to KRR and also applies the kernel trick. But while KRR is trained on linear least squares by applying the L2 regularizer, SVR relies on the ε-insensitive loss. The parameter ε defines an acceptable error margin - the errors smaller than the margin are ignored during training, resulting in a small number of support vectors. Similar as the regularization parameter λ used with LRR and KRR, the error margin ε allows to tune the sensitivity of the model to the independent variables, and hence, to avoid over- or under-fitting of the model. Since the ε-insensitive loss is less sensitive to outliers than the MSE (as used with KRR), SVR is more robust against outliers. Similar as with KRR, we apply SVR with the Gaussian kernel in following parts of this chapter to be capable of modeling non-linear relationships between the features and the outcome.

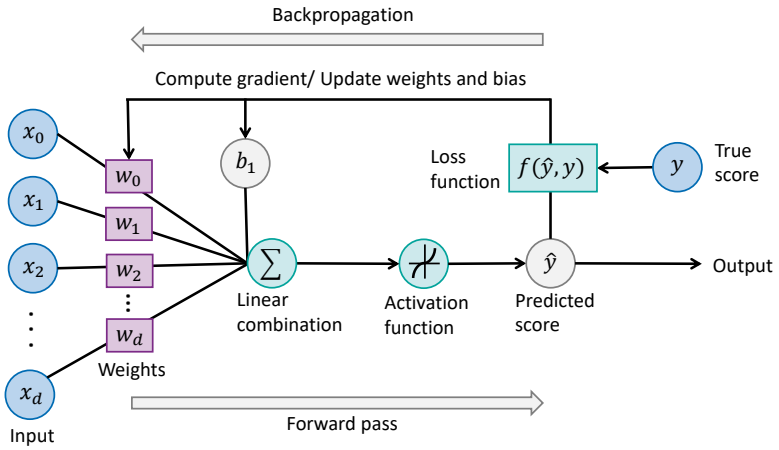


Figure 5.3: Reduced neural network with a single hidden layer for solving regression problems.

5.1.1.5 Neural Networks

NNs [131] are models consisting of artificial *neurons*. A neuron converts a given input to the output by applying a linear transformation with learned weights and then a non-linear transformation, called *activation*. Typically, thousands of neurons form a *layer*, and multiple layers are stacked, where the features correspond to the input for the first layer and the responses correspond to the output of the final layer. The non-linearity of the activation allows to model non-linear relationship between the features and the responses. Actually, NNs with just a single intermediate layer, which has sufficiently many nodes, can approximately express any function. Deep NN with many layers showed excellent performance in many classification and regression tasks, where the raw data, e.g., natural images, without any manual feature engineering are directly fed into the network [133]. This is possible because the first layers work as an automatic non-linear feature extractor, if a deep NN with an appropriate architecture is trained appropriately.

The goal of the evaluations in this chapter is to estimate the QoE on MOS scale, i.e., a continuous number, which thus represents a regression problem. Indeed, NNs are able to pretend to be any type of regression model. Figure 5.3 illustrates a simple neural network, used to solve a regression problem. It consists of one input layer, where each input unit, or neuron, represents a given input, e.g., the specific value of a feature. The inputs are weighted and linearly combined with a bias (denoted as b_1), which is a constant factor to improve the model's fit to the underlying data. The resulting output is forwarded and serves as the input to the *activation function*, which transforms it according to the predefined function. The activation function can be both, linear or non-linear, with sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) being widely used for the latter option. In the shown example, there is only one single hidden layer, because the outcome of the activation function is the predicted score, which represents the output layer. If the NN is still in the training phase, the prediction ($\hat{y}$) is compared against the true value (y) by means of a loss function, which quantifies the prediction error. NNs that are used for regression,

i.e., which predict real-value quantities, typically use the MSE as the loss function. The gradient of the loss function is used to tune the weights of the input layer and the bias. This process is called backpropagation and it improves the network's prediction accuracy with each iteration. Iterations are referred to in the context of NN as *epochs*, whereby one epoch means that an entire data set is passed forward and backward through the neural network once. Another important term is the *learning rate*, which denotes the magnitude of changes to the weights in response to the gradient of the loss function each time the model weights are updated.

5.1.2 Performance Metrics

The performance of ML models with respect to their accuracy can be quantified by comparing the predicted scores against their respective true scores and thus deducing the error. The MSE measures the average of the squares of the errors and is often used during training to optimize the model, e.g., with the L2 regularizer or as the loss function for NN. Apart from that, it is one of the most widely used metrics to assess the estimation accuracy. It computes as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

whereby n denotes the number of samples, y_i the true output of sample i, and $\hat{y_i}$ the predicted output, respectively. The Root Mean Squared Error (RMSE) is accordingly computed as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

and has, compared to the MSE, the advantage to preserve to the unit. For instance, the unit of the output from applying the RMSE is the same as the input units. Another widely used performance metric in the field of ML is the median absolute error (MedAE), computed as

$$MedAE = median\ (|y_1 - \hat{y_1}|, |y_2 - \hat{y_2}|, ..., |y_n - \hat{y_n}|).$$

The advantage of MedAE compared to MSE and RMSE is its lower sensitivity towards outliers. Furthermore, a model's accuracy can be assessed by simply taking the correlations between the predicted output and the true value into account. Two prominent examples to name here, and used throughout our analysis, are Pearson's correlation coefficient (PCC) and Spearman's rank-order correlation (SROCC). While a wide range of further assessment metrics exist [134], we confine in this chapter on the previously presented metrics, which allow a substantial comparison of the different models with respect to their estimation accuracy.

5.1.3 Machine Learning in Communication Networks

The paradigm shift towards softwarized networks and increasingly available network resources leverage the introduction of intelligence to communication systems. ML is deployed in a broad field of use-cases, including traffic classification, routing optimization, resource management, security, and prediction of QoS or QoE. Benefits and drawbacks of different ML approaches in the context of SDN are presented in [135]. This comprehensive survey describes a wide range of algorithms and how they can be used for classification or prediction in today's networks to address problems related to, e.g., privacy and security, or identifying traffic patterns. Similarly, [136] provides a comprehensive body of knowledge on the applicability of ML techniques to support network operations and management. The article focuses on traffic engineering, performance optimization, and network security. ML is widely applied for estimating QoE from network-level KPIs. In this context, many works address video streaming, as this is one of the most prominent applications in today's networks. Due to the ongoing trend towards traffic encryption, ML is often applied on traffic meta-data to estimate QoE-relevant video streaming metrics, such as resolution or bitrate [137, 138, 139, 140], as well as to predict [141] or classify [142, 143] the QoE in terms of the MOS.

Mobile video streaming is getting more and more popular. Due to additional network-related KPIs, such as the channel quality, and additional characteristics of clients, e.g., their movement, QoE assessment in mobile environments needs dedicated evaluations. Therefore, [144] focuses on mobile networks when studying the performance of various classifiers and the impact of the used features retrieved from network- and application-related data. However, it is desirable to estimate the QoE solely using network-related features, as such data is typically available to an MNO or ISP and easier to obtain at scale than detailed application-level information. Such solutions, where the estimation is solely based on network-level KPIs, are presented for LTE networks in [145] and for 5G networks in [6]. Both approach assume to have network data available on a very fine-grained basis, which are generally harder to obtain. To overcome this issue, [146] analyzes how TLS transactions can be used by ISPs to estimate video QoE from coarse-grained data. A first industry solution which utilizes ML in combination with network, client, and codec information for the assessment of the network's impact on speech quality for VoIP services is sQLEARN[1] from Infovista. It is intended for 4G/LTE and as well for 5G networks, both carrier, i.e., VoLTE, and OTT services.

With the introduction of 5G, new opportunities and challenges arise for ML-driven network operations. The work in [147] focuses on the potential solutions for 5G from an ML perspective. The authors discuss some promising approaches on how ML can contribute to support 5G and evaluate the impact and possible limitations of such approaches. Thereby, QoE is often mentioned as a possible use-case for ML in 5G. While [148] focuses on the challenges and proposes a data-driven architecture for QoE management, the work in [149]

[1] https://www.infovista.com/sites/default/files/resources/WP_Infovista_Learn_About_sQLEAR_in_Motion.pdf.pdf

goes a step beyond the QoE assessment by presenting an ML-based resource allocation for 5G networks. The proposed system determines the network performance level via clustering, predicts network KPIs by means of regression, and dynamically provisions resources in a proactive way. Similarly, the work in [150] applies ML techniques for traffic forecasting to efficiently scale 5G core network resources and [151] proposes to use ML to predict the number of users in a cell so to optimize traffic routing. One of the new key features of 5G, network slicing, is exploited in conjunction with deep learning [152] and reinforcement learning [153] to achieve an optimized resource utilization. Both are similar to DeepCog [154], an approach using neural networks to forecast the resource demands of network slices so to perform an appropriate resource allocation.

The exploitation of the newly introduced NFs in 5GS for intelligent networking has recently been proposed in literature. For example, it is examined how the NWDAF can be used for predicting abnormal as well as expected behavior for a group of UEs, and to forecast the network load in an area of interest. The proposed architecture connects the NWDAF with other NFs via the SBI to allow mutual data transfer. Using both, time series data and generated features available at the NWDAF, different ML models are examined with respect to their feasibility for the given problems. The conducted study shows that NNs outperform LRR models when it comes to network load prediction and that tree-based XGBoost yields better classification performance compared to logistic regression in the anomaly detection use-case. Due to the huge amount of devices expected to be connect to 5GS, moving all data to a centralized unit for analytics is un-efficient. The need for running ML algorithms in a distributed manner is discussed in [155]. Only then, a fast decision making which minimizes the network response time to user requests, and to fulfill the latency requirement of 5G, can be guaranteed. The outlined proposal for a distributed analytics architecture considers one centralized NWDAF instance and several distributed NWDAF instances, which can be collocated with other NFs and only collect data gathered from that collocated NF.

5.2 Challenges and Design Criteria from MNO Perspective

An MNO planning to integrate ML-based QoE estimation into its network is faced with a variety of different challenges and design decisions. As there is no one-fits-all solution, we first elaborate on the major questions to be addressed in order to tailor such an integration to an MNO's needs and describe afterwards how we exemplarily account for them in the scope of this chapter.

Q-1 **Can the QoE be estimated using ML and solely based on network KPIs?**: While ML is a promising approach to overcome the limitations of QoE monitoring in today's networks, the relationship between different metrics which can be monitored in the network and the QoE is complex. Answering the question how reliable and efficient such a QoE estimation can be carried out, consequently forms a crucial prerequisite.

Q-2 **How to estimate the QoE in 5G mobile networks?**: This question relates to the specific realization implemented by the MNO for the ML-based QoE estimation and it inherits a couple of follow-up questions.

Q-2.1 **How to efficiently exploit new features provided in 5G networks?**: The 5G architecture provides new NFs dedicated for enhanced data analytics (NWDAF) and for third party information exchange (AF). Their specific usage is however not part of 5G standardization activities and it resides with the MNO to determine how to particularly exploit their capabilities.

Q-2.2 **Which features need to be provided at the NWDAF for training and testing and how to obtain them?**: While the network-related data, which is later used for the estimation, is easily accessible to the MNO, the application-related data required to obtain the QoE is typically only available to the AP. Hence, one of the main challenges is to settle a proper way for obtaining both types of data. The collected data then needs to be analyzed and pre-processed to make it useful for later usage. It needs to be ensured that both sources of data, i.e., network- and application-related data, provide information addressing the same target, e.g., the same time period, the same geographical area, or the same access technology. First analysis of the ground-truth data should include the granularity of available data (i.e., are time series available per second, per minute, or per session) and provide insights regarding the relationship between the monitored data and QoE, to assess a data source's relevance for the estimation process.

Q-2.3 **Which ML models should be considered for potential deployment?**: The field of ML offers a huge amount of individual techniques. They differ with respect to their training process, i.e., supervised versus un-supervised learning models, as well as in terms of their respective prediction target, which can be clusters of given data points, a categorization of input samples into a pre-defined set of classes, or actual numbers covering a continuous range. Furthermore, the available models range from very simple approaches, such as linear regression techniques, to highly advanced and complex approaches like artificial neural networks. Which one to choose is a crucial design criteria and depends on numerous factors.

Q-2.3.1 **Which specific use-case should be addressed?** The purpose of the QoE estimation determines the requirements and hence narrows down to a specific set of potential ML algorithms for deployment. Different use-case have different demands, e.g., in terms of the estimation speed or the level of estimation granularity. If only a rough estimation of the system's QoE on a large time scale is of interest, it can be sufficient to run simple classification algorithms. If the QoE estimation should be used for a QoE-aware, real-time resource control on a per-flow level, the requirements in terms of estimation speed, accuracy, and granularity are significantly higher.

Q-2.3.2 **How well does an algorithm perform?**: The estimation accuracy is one of the key factors for determining the algorithm to deploy. However, while a high estimation accuracy is in general favorable, it might not be the ultimate goal. For instance, an MNO might still deploy another model than the best performing one, if it is capable of better meeting the overall trade-off, which can, for example, involve factors such as the comprehensibility or robustness towards over-fitting.

Q-2.3.3 **What are potential influence factors on the accuracy?**: Today's networks are very heterogeneous in many terms. It needs to be examined if, and to which extent, different factors impact the estimation accuracy. This includes for example the movement patterns of users or the service type, for which the QoE should be estimated.

Q-2.3.4 **Deploying a one fits-all solution or dedicated algorithms for different services?**: If the same ML algorithm can be used for all applications, the MNO can benefit from reduced costs for maintenance and manpower. If a dedicated algorithm should be used for each specific service, this can be beneficial in terms of estimation performance.

Q-2.3.5 **How comprehensible is the model?** This design decisions refers to whether the MNO aims at understanding what lead the algorithm to the specific output, or if the black-box behavior, coming along with some ML models, can be accepted.

Q-2.3.6 **How costly is a model in terms of resources?**: Besides the accuracy and the comprehensibility, an MNO should also take the computational and temporal overhead into account to ensure that the mechanisms scale and can be applied efficiently to not overload the responsible NF. For instance, the MNO needs to be aware of the exploitation of available resources during training and testing, as well as of the duration until an estimate is available.

Q-2.4 **How to maintain deployed models?** The Internet ecosystem is highly dynamic, leading to the fact that a once trained and deployed model will loose its validity with ongoing changes in the system. For example, the usage of a new voice codec in a VoIP service will affect the relationship between the measured network KPIs and the resulting QoE and hence differ from the associations as learned by the model. Accordingly, it is advisable, or even mandatory, for the MNO to still collect ground-truth QoE data for updating and re-validating a deployed model. By means of regular sanity checks and re-trainings, the MNO needs to ensure that the model is still adequate, or deploy an updated version to account for system changes. Consequently, a deployed model can evolve over time, which potentially raises another challenge if the MNO aims to keep track of the evolution of the model.

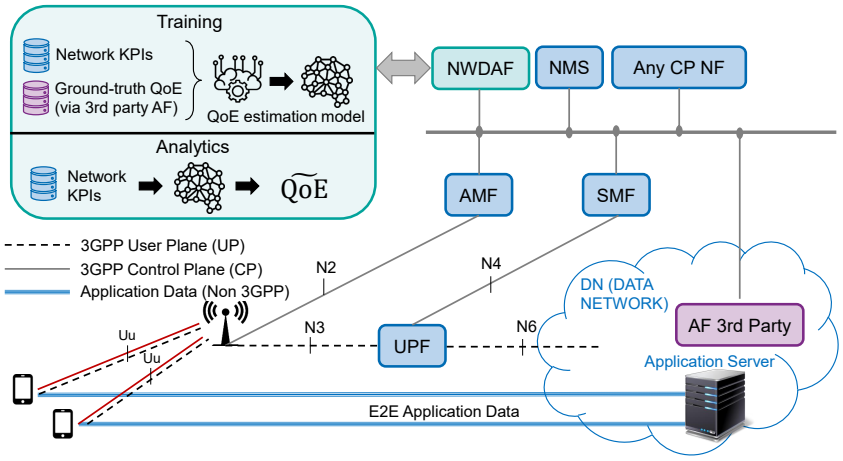


Figure 5.4: Integration of AF and NWDAF in the 5G architecture to support data analytics and third party information exchange.

Elaborating on these challenges allows an MNO to analyze the trade-offs and find the sweet spot in the variety of design goals, which can exclude each other. For example, the design goal of deploying a simple model can conflict with the demand of a high estimation accuracy, if the complex models outperform the simple alternatives. Besides accuracy and complexity, other criteria which need to be taken into account are related to the costs and the robustness of algorithms, e.g., in terms of the used application.

5.2.1 Data Analytics-driven QoE Estimation in 5G Networks

Based on Figure 5.4, we describe two newly introduced NFs, which are relevant for our work. The first one is the Application Function (AF). It is a 5G core network function, which can be owned and customized by third parties, and hence enables communication between 5G core control plane NF (owned by the MNO) and content or application providers, e.g., YouTube or Netflix. It provides a standardized interface, connected to the Service Based Interface (SBI), which allows for sharing information related to the application with any other 5G core control plane NF.

The second one is the Network Data Analytics Function (NWDAF). This NF is also connected to the SBI and capable of collecting and processing statistics from other 5G control plane NFs, i.e., the Access and Mobility Management Function (AMF), Session Management Function (SMF), or the Network Management System (NMS). Besides, the NWDAF can also obtain data from third parties via the AF, e.g., information about the QoE. As the entity where all information is accumulated, the NWDAF can be seen as the brain of the 5G system, which provides intelligent services to all NF by exposing analytics, which other NFs can invoke and where ML algorithms may run. Dedicated for an ML/AI deployment, the NWDAF can be decomposed into two logical functions: *training* and *analytics*. The NWDAF which holds the logical function for training takes care for the development of the model, including the initial training and regular re-training of the model with new training

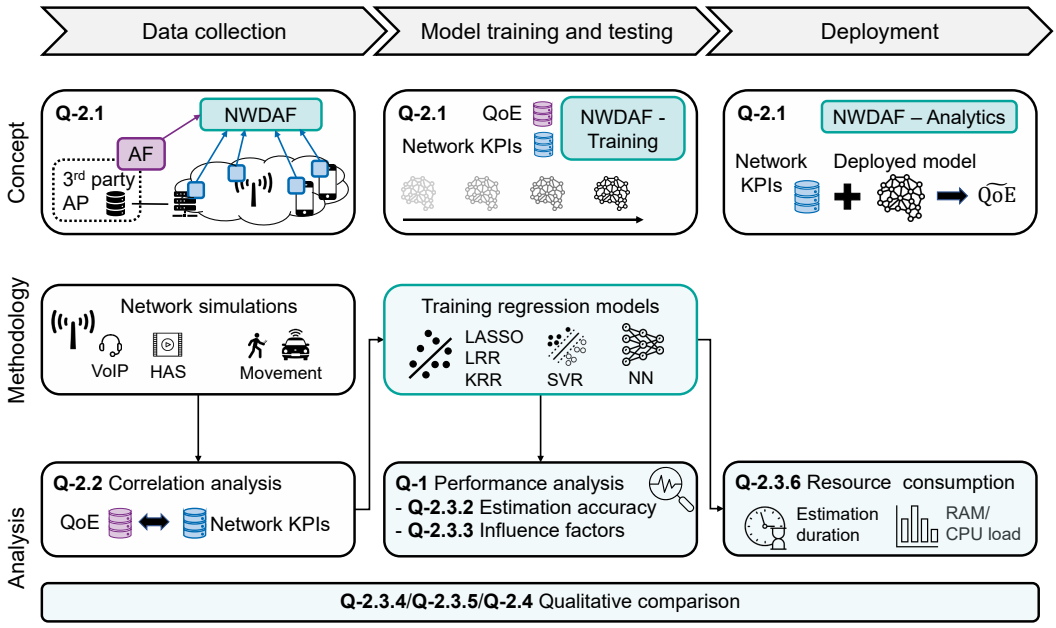


Figure 5.5: Contributions of this chapter to address the challenges of ML-based QoE estimation, with respect to the three steps (data collection, model training and testing, and deployment). The notation Q-X specifies the respective question which is addressed. Light green boxes denote elaborations carried out on the example of the five regression techniques.

data, to keep it up to date. The NWDAF containing the analytics function applies the trained models, i.e., it estimates the QoE.

For the scope of this chapter, we assume the following workflow: The AP, which is aware of any application-related KPI, provides the QoE scores via the AF, e.g., retrieved via the P.1203 model. The MNO, which can collect any network-related KPI at any NF connected via the SBI, statistically processes the monitored data at the NWDAF, i.e., it generates features from the monitored data. The logical function of the NWDAF dedicated for *training* is applied to learn the relationship between the generated features and the QoE. Once the model is capable to obtain a satisfactory estimation accuracy, it can be deployed via the NWDAF's logical *analytics* function, allowing the MNO to assess the QoE also in the absence of information provided by the AP via the AF. Such a mechanism consequently allows the MNO to perform QoE monitoring in its network.

5.2.2 Conducted Feasibility Study

By means of a feasibility study, we demonstrate the applicability of the approach described above and show exemplarily how the different challenges and design criteria can be addressed. Figure 5.5 classifies our contributions with respect to thee three major phases *data collection*, *model training and testing*, and *deployment*. The top of the illustration denotes

our proposed concept, exploiting the 5G NFs for an ML-driven QoE estimation, thus focusing on Q-2.1. The blue squares denote the network KPIs obtained during the *data collection phase* at different instances, e.g., at the User Equipment (UE) or the Access Node (AN), which are statistically processed to generate features. Additionally, during this phase, true QoE scores are provided via the AF (denoted as the purple box) from a third party AP. In the course of our practical study, we gather these types of data by means of network simulations and for two exemplary applications, i.e., HAS and VoIP. Based on the generated data set, we to address Q-2.2 by studying the correlation between the network-related features and the QoE.

During the *model training and testing phase*, the logical NWDAF function for *training* learns the relationships between the network KPIs and the QoE. For our feasibility study, we assume that the MNO's goal is to estimate the QoE on MOS scale. Accordingly, we train regression techniques, which are capable for predicting any number in the given range between 1 and 5. While this is more complex compared to a classification into pre-defined classes, e.g., low, medium, and high QoE, the finer granularity of the prediction output allows to cover a wider range of possible use-cases, including reporting, on-demand troubleshooting, or automatic corrective actions. We use the least absolute shrinkage and selection operator (LASSO), Linear Ridge Regression (LRR), Kernel Ridge Regression (KRR), Support Vector Regression (SVR), and Neural Networks (NN). Although the set of used techniques is relatively small, it is representative in terms of the models' complexities. Our performance analysis addresses Q-1 by demonstrating the general feasibility of estimating the QoE from network KPIs, as all trained models yield reliable outputs. Besides the general comparison with respect to the estimation accuracy (Q-2.3.2), we study in how far it is influenced by a user's movement pattern, the obtained QoE, or the used application (Q-2.3.3).

According to the proposed concept, during *deployment*, the logical function of the NWDAF which is dedicated for *analytics*, holds a trained ML model. Thus, based on the network KPIs, it can estimate the QoE. In the scope of our feasibility study, we analyze the resource consumption, as well as the time it takes the different regression techniques to obtain the QoE estimates, thus addressing Q-2.3.6. Finally, we compare them in a qualitative manner, considering factors such as their comprehensibility, trackability, or whether they provide built-in features supporting over-fitting prevention or feature selection, to account for Q-2.3.4, Q-2.3.5, and Q-2.4.

5.3 Evaluation Methodology

This section introduces the applied methodology for our studies. We first detail on the conducted simulations to generate the ground-truth data set, along with the considered applications and scenarios. In a next step, we describe the procedure for training the different regression techniques.

5.3.1 Simulation Scenarios for Ground-truth Data Collection

For generating the network- and application-related ground-truth data, we rely on the discrete event simulator OMNeT++ [156]. Additionally, we use the frameworks INET and SimuLTE. Although SimuLTE simulates 4G networks, the type of monitored information is the same with 5G networks. For instance, the equivalent of 4G Packet Gateway (PGW) and eNB data is information collected from UPF and gNB in 5G. Four our feasibility study, the main point of using SimuLTE is to obtain monitoring information from both, access and core network. As we assume the monitoring information to be available at NWDAF and do not consider any signaling exchange for data collection, SimuLTE can be used for our purpose of generating user plane traffic in a mobile network. From the perspective of the radio technology, 5G has much higher data rates compared to 4G. However, the principles of system load (number of UEs in a cell) and radio quality will still play a role in 5G systems. Our simulations consist of a single AN which serves a varying number of active UEs, which differ with respect to their mobility characteristic. On the one hand, we consider static clients, which do not move throughout the experiments. Some clients, on the other hand, move between different points of interest (POIs) within the cell. To simulate realistic movement patterns, we take the small world in motion (SWIM) movement model [157] into account. Moving clients are either considered as pedestrians or vehicles, with speeds of 3 kmph and 50 kmph, respectively.

For the active clients, we consider two different types of applications. The first one is VoD application, according to the HAS principle. The video streamed by the clients has an overall duration of 400 seconds, split into small segments of 5 seconds each, and is made available in four different qualities, comprising bitrates of 500 kbps, 1 Mbps, 1.5 Mbps, and 3 Mbps. The VoD client applies a buffer-based heuristic, hence, the segment quality is determined based on the video buffer's filling level. The lowest quality level is chosen if the buffer contains less than 10s of video play time. The second, third, or fourth quality level is chosen if the buffer exceeds the threshold of 10s, 20s, or 30s, respectively. In order to make our results more generic, and not limited to a specific QoE model, we apply two different existing models to obtain the QoE for VoD clients. The first one is the ITU-T P.1203 model, the second one is the cumulative quality model (CQM). In the following, we will refer to these models as *VoD-P.1203* and *VoD-CQM*. The QoE is computed on a per-session level, i.e., after the 400s of the video clip have been played back. CQM allows to set different parameters, such as weights for different quality window metrics and we use the default values from the implementation's repository[2].

The second application we consider is VoIP, for which we only model the receiver side, i.e., the listener of a conversation. The talk spurt duration follows a Weibull distribution with a scale of 1.4 seconds and a shape of 0.82 seconds. Between the talk spurts, there is silence, which also follows a Weibull distribution (0.899s, 1.089s). We use the g726 codec and a bitrate of 40 kbps. Although better voice coder exist, the voice processing in the example under study does not have to be representative of reality, but at least allow quality variation

[2] `https://github.com/TranHuyen1191/CQM`

all along the available range, so that training and testing data can be used to compare the ML techniques under study. The QoE of the VoIP client is computed using the e-model as defined in the ITU-T G.107 standard [158]. Thereby, a MOS value is computed after each talk spurt. To obtain the clients' overall QoE for the whole VoIP session, we take the average MOS of all talk spurts. Newer models, such as the ITU-T P.863 (POLQA) are capable to better reflect the user's perceived voice quality. POLQA is a FR metric, i.e., it needs a reference audio sample to compare it against the received audio signal to obtain the MOS. However, due to the simulative nature of our approach, such FR metrics cannot be applied in our case. Nevertheless, the general relationship between MOS and its impacting factors is the same for both models [159, 160]. Hence, we can expect a similar performance of the applied ML models, independent of the applied QoE model.

To obtain QoE values exploiting the MOS range as good as possible, we run simulations with a slight overload of the cell's capacity. For VoD, we consider 80 and 160 active clients. When running simulations for VoIP, we increase the number of clients to 400, as a VoIP client consumes significantly less bandwidth. The clients are either moving or static and we set the following distribution with regard to the UEs' mobility patterns: 100% static, 100% moving, or half moving half non-moving. To vary the impact of the clients' movement patterns on network KPIs, we configure four different POI settings: A single POI, which is located at the edge of the cell or a single POI, which is close to the AN. Next, we consider 10 and 100 POIs, which are randomly placed within the cell. Finally, we simulate each configuration with different seeds, which determine the initial placement of the UEs.

5.3.2 Collected Monitoring Information and Features

With the term *ground-truth* data, we refer to those data points, where we have network-related information and its associated true QoE expressed as MOS. While the estimated QoE is the output of the trained ML models, the true QoE is reliably retrieved by applying t the QoE models. To obtain the true QoE, we collect all relevant network-related and/or application-related information needed with the respective QoE model, e.g., video quality and interruptions in the case of VoD. The QoE estimation performed by the ML models, however, solely relies on network-related information. Our monitored network information for the different application types are summarized in Table 5.1. All of them are collected as time series. To generate the features, which are the input for the ML models to estimate the QoE, we apply the 12 different statistics shown in the table to the time series. Please note that in the case of VoIP, where we only consider the transmission from sender to receiver, there is no up-link related monitoring information. The only exception is the AN throughput, whose upload throughput reflects the sum of the download throughput of all active UEs. We furthermore collect two different types of delays in the case of VoIP: the end-to-end delay and the Radio Link Control (RLC) delay. The end-to-end delay is an in-app measurement and possibly not available to an MNO. However, the MNO is in any case capable to obtain the RLC delay, which is, in our case, monitored on a per-UE basis at the AN.

Table 5.1: Overview of the data monitored in the network and the applied statistics to generate network-level features. Check marks denote that the monitoring information is collected for estimating the QoE of the specific application.

Network KPI	**Abbreviation**	**VoD**	**VoIP**
Access node throughput on uplink	AN TP UL	✓	✓
Access node throughput on downlink	AN TP DL	✓	✗
User equipment throughput on uplink	UE TP UL	✓	✗
User equipment throughput on downlink	UE TP DL	✓	✓
Channel quality indicator on downlink	CQI DL	✓	✓
Channel quality indicator on uplink	CQI UL	✓	✗
TCP Round trip time	RTT	✓	✗
End to end delay	E2E delay	✗	✓
Radio link control delay	RLC delay	✗	✓
Hybrid automatic repeat request error rate on downlik	HARQ ER DL	✗	✓
Applied Statistics	**Abbreviation(s)**		
Average / minimum / maximum / median	avg / min / max / median		
Standard deviation/ variance	std / var		
25th percentile / 75th percentile	25perc / 75perc		
Coefficient of variation / kurtosis / skewness	cov / kurt / skew		
Unbiased standard error of the mean	sem		

5.3.3 Training of Machine Learning Models

We split the set of ground-truth data points into a *training set*, which comprises 80% of the data, and a *test set* for the remaining 20%. The training set is used for training the ML models and tuning their hyper-parameters, i.e., those parameters which are not set upfront, but optimized during the training phase. During training, we apply a 5-fold cross-validation, i.e., the training is repeated 5 times. Thereby, 80% of the *training set* samples are used for the actual training and 20% to validate the trained model's performance. In each of the 5 rounds of training, the set of 20% of samples used for the validation changes. That way, each sample in the *training set* is used four times to train the model and is used once to validate a trained model's performance. In each of the five rounds, a *validation error* and a *training error* are obtained. Both types of errors need to be considered to ensure that a model does not under- or over-fit. A high validation error in combination with a low training score indicate an over-fit of the model, while a high validation error in combination with a high training error reveal an under-fitting model. The sweet spot is located where the validation error is low, with a moderate training error. The hyper-parameter set which maximizes the validated performance (averaged over the 5 folds) is chosen for the final model. Hence, the hyper-parameters are tuned and the trained model can be applied to the *test set*. Contrary to the validation set, the *test set* (the 20% of the whole ground-truth which has not been used during the training phase) only comprises data that the ML model has not seen so far, i.e., none of the samples of the test set were used to train the model. By applying the model to the test set, we obtain the *test error*, which is used for reporting the final performance. Consequently, when evaluating a model's accuracy, we are referring to the *test error*.

Table 5.2: Parameter settings for training the ML algorithms with VoD. The selected parameters are those which yield the highest estimation accuracy among the tested parameters and which are used in the evaluation chapter.

	Parameter	Description	Tested	# comb.	Selected
LASSO	λ	Regularization	$[0:0.1]$ ($\|\lambda\| = 21$)	21	0.0005
LRR	λ	Regularization	$[0:500]$ ($\|\lambda\| = 28$)	28	10^{-9}
KRR	λ	Regularization	$[0:10]$ ($\|\lambda\| = 23$)	23	0.07
SVR	C	Regularization	$[0.1:6]$ ($\|C\| = 16$)	32	4.92
	ε	Acceptable error	$[0.01, 0.1]$		0.01
NN	*LR*	Learning rate	$[0.001, 0.01, 0.1]$	504	0.001
	#epochs	Number of epochs	$[100, 200, 500, 1000]$		1000
	#neurons	Number of neurons	$[10, 50, 200]$		50
	batchS	Batch size	$[10:1000]$ ($\|batchS\| = 7$)		200
	AF	Activation function	tanh, relu		relu

Table 5.3: Parameter settings for training the ML algorithms with VoIP. The selected parameters are those which yield the highest estimation accuracy among all tested parameters and which are used in the evaluation chapter.

	Parameter	Description	Tested	# comb.	Selected
LASSO	λ	Regularization	[0:0.1] ($\|\lambda\| = 21$)	21	0
LRR	λ	Regularization	$[0.1:600]$ ($\|\lambda\| = 28$)	28	0.1
KRR	λ	Regularization	$[0:1]$ ($\|\lambda\| = 23$)	23	0.01
SVR	C	Regularization	$[7:12]$ ($\|C\| = 16$)	32	12
	ε	Acceptable error	$[0.01, 0.1]$		0.03
NN	*LR*	Learning rate	$[0.001, 0.01, 0.1]$	504	0.01
	#epochs	Number of epochs	$[100, 200, 500, 1000]$		1000
	#neurons	Number of neurons	$[10, 50, 200]$		50
	batchS	Batch size	$[10:1000]$ ($\|batchS\| = 7$)		200
	AF	Activation function	tanh, relu		relu

All ML algorithms are tested using various settings for their respective parameters. Table 5.2 summarizes the tested settings for each ML option when training for the application VoD, and Table 5.3 when training for VoIP, respectively. For the NN, we consider a single hidden layer. The table further denotes those parameter settings, that have been chosen for the QoE estimation for the different service types. The chosen parameters are those which yield the highest estimation accuracy, i.e., the lowest validation error expressed as MSE. These parameter settings are used when comparing the different algorithms in the evaluation chapter.

Table 5.4: Number of samples and average MOS ± standard deviation for the different subsets of the ground-truth data set.

	VoD-P.1203		*VoD-CQM*		*VoIP*	
Data set	# samples	MOS	# samples	MOS	# samples	MOS
all	13952	2.18±0.60	13952	1.65±0.79	27592	4.17±0.49
stationary clients	3210	2.41±0.82	3210	1.97±0.98	8392	4.04±0.67
moving clients	10742	2.12±0.49	10742	1.55±0.70	19200	4.23±0.37
low QoE	8095	1.83±0.07	10139	1.89±0.13	291	1.21±0.24
medium QoE	5067	2.46±0.39	3295	2.68±0.35	2136	2.97±0.40
high QoE	790	4.02±0.36	518	3.74±0.16	25165	4.31±0.18

5.4 Ground-truth Data Set Analysis

This section presents the characteristics of the ground-truth data set. We first detail on the number of collected samples and show the distributions of the ground-truth QoE for VoIP and VoD. Next, we study the correlations of the network-related features with QoE and exemplary show the impact of the most powerful features' values on the obtained MOS scores.

5.4.1 Ground-truth QoE Distribution

In order to allow more fine-grained analysis, we divide our ground-truth data set into subsets. Thereby, we distinguish between *stationary* and *moving* clients. We further categorize the data samples according to their true QoE score into three classes: *low*, if the QoE is below 2.0, *medium*, if the QoE ranges between 2.0 and 3.5, and *high* if the QoE is above 3.5. We refer to to the overall data set as *all*. We will refer to these subsets throughout our study, to evaluate the performance for different ranges of the QoE and varying movement patterns of the clients. Table 5.4 denotes the number of ground-truth samples for each service type in the different subsets and it shows the average MOS values along with their standard deviations. The average QoE of the VoD clients is higher when using the ITU-T P.1203 model (2.18), compared to using CQM (1.65). With both models, *stationary* clients obtain a better QoE than the *moving* clients. For moving clients, we observe a lower standard deviation than for stationary clients. This can also be observed for VoIP, where moving clients have a standard deviation of 0.37, while for the stationary ones, the standard deviation increases to 0.67. Contrary to VoD, moving VoIP clients obtain a slightly higher QoE than stationary ones. This is counter-intuitive, since mobility is a well known degradation factor in telecommunications. This issue nevertheless does not pose a problem in terms of training and testing the ML models, as the global distributions of MOS scores, and hence the training of the models, are not biased.

The QoE distributions for the different services are denoted in Figure 5.6. The x-axis denotes the ground-truth QoE scores, the y-axis denotes the ECDF. While for the VoD clients, about 60% (P.1203) or 70% (CQM) suffer from low QoE, about 90% of the VoIP clients

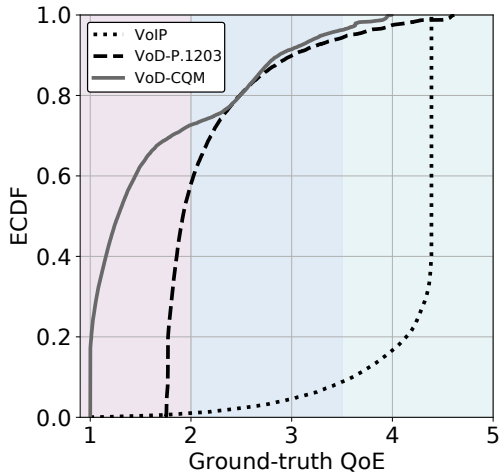
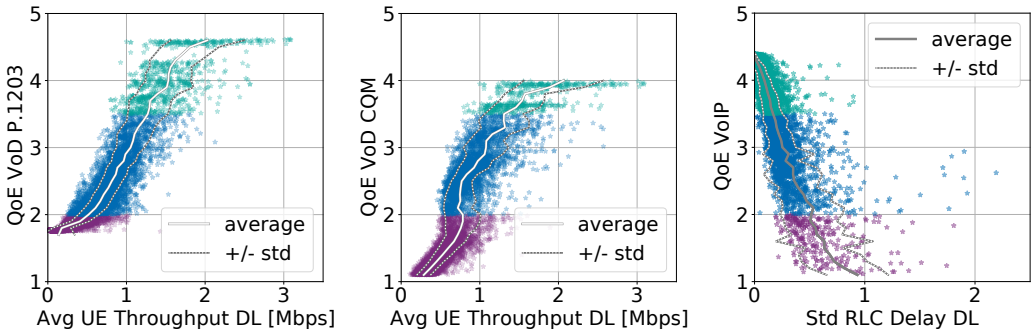


Figure 5.6: Ground-truth QoE scores obtained for VoIP and for VoD with the different QoE models.

achieve a high MOS score. We want to add here that the data sets are not equally distributed in terms of the true QoE. The low QoE scores for the VoD clients can be explained with the highly increased bandwidth requirements of VoD in the specific context of the simulated environment used during this study. Although the ground-truth QoE distribution could easily be harmonized by removing samples of too frequent MOS values, we use the data set as it is. An MNO who actually deploys ML in its system will face similar issues. Additionally, it allows us to study the estimation accuracy in a more natural/realistic setting.

5.4.2 Relationship Between Most Expressive Feature and QoE

Next, we study the relationship between the QoE for VoD and VoIP clients and their respective features with the highest correlation according to Pearson's correlation coefficient. For VoD, independent of the applied model, the feature with the highest correlation with QoE is the average downlink throughput. We first detail on this feature's impact on the VoD QoE when computed according to the P.1203 model. Figure 5.7a denotes the average UE downlink throughput on the x-axis and the QoE scores on the y-axis. The white line shows the average values of the UE donwlink throughput obtained within discrete MOS intervals with steps of 0.1. For example, it denotes the average UE donwlink throughput of all clients which achieve a MOS between 3.5 and 3.6. Additionally, we denote with the dotted lines the average value +/- its standard deviation. We can observe a linear behavior between the average throughput and QoE. As expected, the higher the average throughput, the higher the MOS score. The plot further shows that with increasing MOS scores, the standard deviations increase and there are more throughput values that lie far away from the average of the 0.1-sized QoE interval. Figure 5.7b accordingly shows this relationship for the VoD QoE as computed by CQM. As expected, the relationship between the average downlink throughput and QoE is similar as with VoD-P.1203.



(a) VoD P.1203. **(b)** VoD CQM. **(c)** VoIP.

Figure 5.7: Relationship between the QoE and the respective most expressive network-related feature.

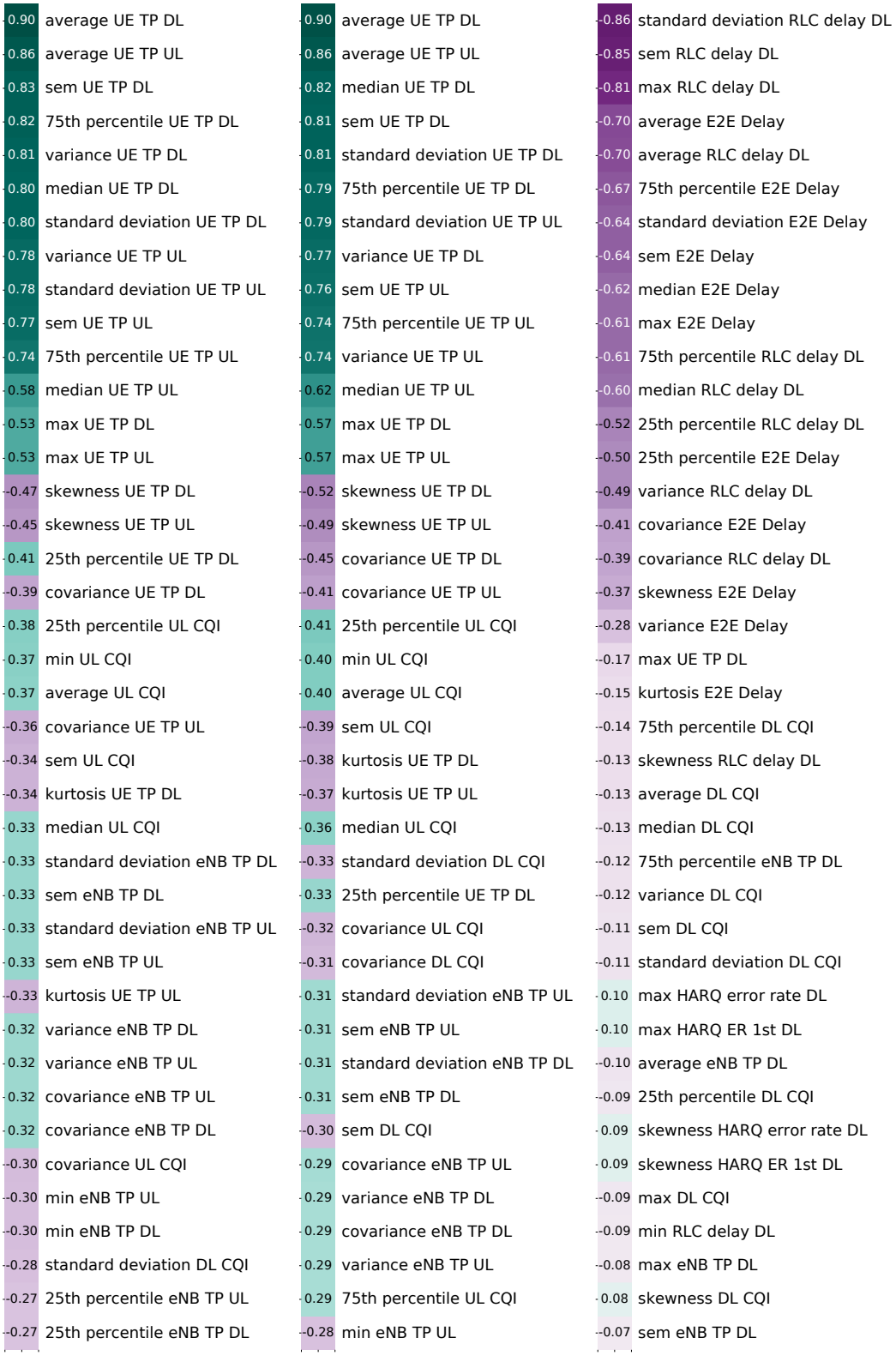
For VoIP, the feature with the highest correlation to QoE is the standard deviation of the downlink RLC delay. The impact of this feature's values on the MOS scores is denoted in Figure 5.7c The higher the RLC downlink delay, the lower the MOS. Contrary to the VoD clients, the standard deviation of this feature's value increases with decreasing MOS values. Furthermore, we can see several outliers, especially in the region of medium and low QoE.

5.4.3 Feature Correlation Analysis

Figure 5.8 denotes the correlation between the QoE score and the different features according to PCC. For the sake of clarity, we confine on the 40 top ranked features. While green boxes denote a positive correlation, purple colored boxes denote a negative correlation, respectively. For the VoD QoE according to P.1203, the highest correlation observed is the average UE downlink throughput (0.90), followed by the average UE uplink throughput with a value of 0.86. The subsequent 16 features are all related either to the UE uplink or downlink throughput, showing the high relevance of this monitoring statistic for the QoE estimation. The first feature which is related to a different monitoring metric than UE throughput is the 25th percentile of the UE uplink CQI, with a correlation coefficient of 0.38, followed by the minimum and average uplink CQI (0.37).

The correlations when using CQM differ only slightly. Same as with P.1203, the two highest correlations are obtained for the average UE throughput on downlink (0.90) and uplink (0.86). Again, the first feature not related to UE throughput is the 25th percentile of uplink CQI, which has a slightly higher correlation of 0.41 when using CQM instead of P.1203.

The most expressive feature in case of VoIP is the standard deviation of the RLC downlink delay with a correlation of -0.86. In general, features generated from the RLC delay are highly correlated with the VoIP QoE. The first feature which is not related to any delay metric is ranked on the 20th place. It is the maximum UE downlink throughput which has a comparably low influence on the QoE with a correlation coefficient of only -0.17.



(a) VoD P.1203. **(b)** VoD CQM. **(c)** VoIP.

Figure 5.8: Correlation between network-related features and QoE. Only the 40 features with the highest PCC scores are shown. Values are sorted from top to bottom according their absolute correlation values.

5.5 Performance Evaluation

This section evaluates and compares the different regression techniques considered for this chapter. We first perform a quantitative assessment and study the QoE estimation accuracy. Next, we investigate the different mechanisms with respect to their resource requirements and the duration for training and testing the models. Finally, as an operator might also consider different non-qualitative aspects when deciding about the algorithm to deploy, we compare the different mechanisms with respect to different qualitative factors.

5.5.1 Quantitative Assessment

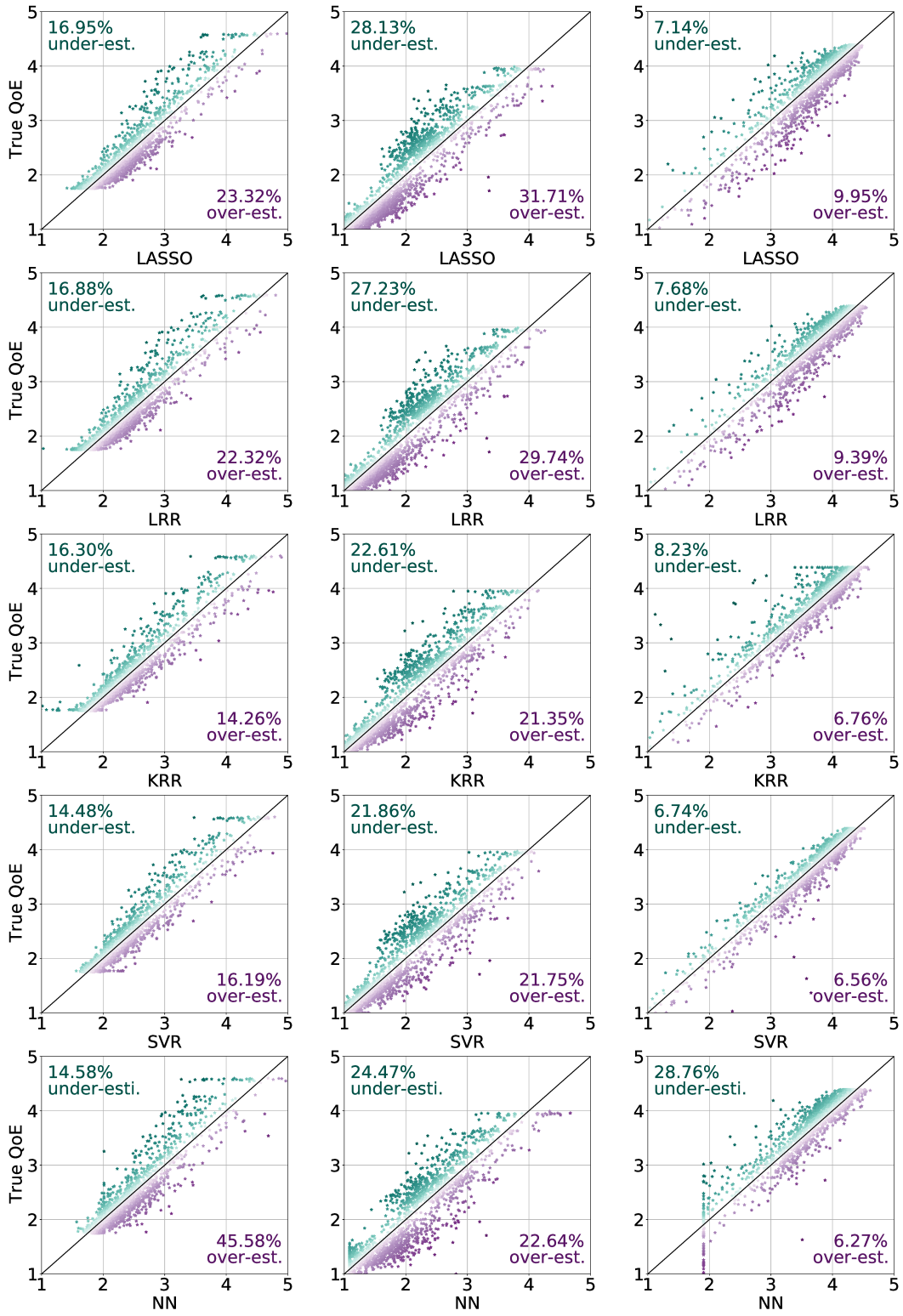
For assessing the different mechanisms quantitatively, we first take a detailed look on the estimated versus true QoE. In a next step, we investigate how the estimation accuracy differs for the different data sets, i.e., depending on a client's movement characteristic or true QoE. The quantitative assessment concludes with an investigation of meta KPIs, such as the duration or CPU load for running the mechanisms.

5.5.1.1 Estimated versus True QoE

This section studies the deviation of the estimated MOS score from the true score. Thereby, we also evaluate whether a mechanisms tends to over- or under-estimate the QoE. To do so, we assume an estimation to be accurate, if it deviates less than 0.1 from the true score. For instance, if the true QoE is 2.7, any estimated value between 2.6 and 2.8 is seen as accurate. Values lower than 2.6 represent an under-estimation and higher than 2.8 an over-estimation, respectively.

Figure 5.9a illustrates the true QoE and its estimation from the five regression techniques for VoD when using the ITU-T P.1203 model. The angle bisector represents the cases where the estimation equals the true QoE, i.e., the optimal case. Values above this line are under-estimations (shown in blue), values below this line are over-estimations (shown in red). All of the tested mechanisms have a similar fraction of under-estimates. The lowest rate is obtained for SVR (14.5%) and the highest one for LASSO, with about 17%. More significant differences can be observed when it comes to the QoE over-estimates. Thereby, KRR outperforms the other mechanism with a fraction of about 14% over-estimation. NN performs the worst and over-estimates the QoE in nearly half of the test samples.

When the true VoD QoE is obtained with CQM instead of P.1203, the accuracy decreases for all of the five regression techniques, as shown in Figure 5.9b. Similar as with P.1203, SVR and KRR yield the lowest fraction of under-estimations. However, the fractions are increased by roughly 7% compared to P.1203, resulting in an under-estimation rate of about 22% for SVR and 23% for KRR. The over-estimation rate increases as well when using CQM instead of P.1203. The only exception is NN, which over-estimates less for CQM than for P.1203. Another observation that can be drawn compared to P.1203 is the increased



(a) VoD P.1203. **(b)** VoD CQM. **(c)** VoIP.

Figure 5.9: Accuracy of the different regression techniques for the different service types. Deviations of estimated QoE larger than +0.1 are counted as over-estimation, deviations larger than -0.1 are counted as under-estimation. Estimated values within the +/- 0.1 boundary are assumed as accurate estimations.

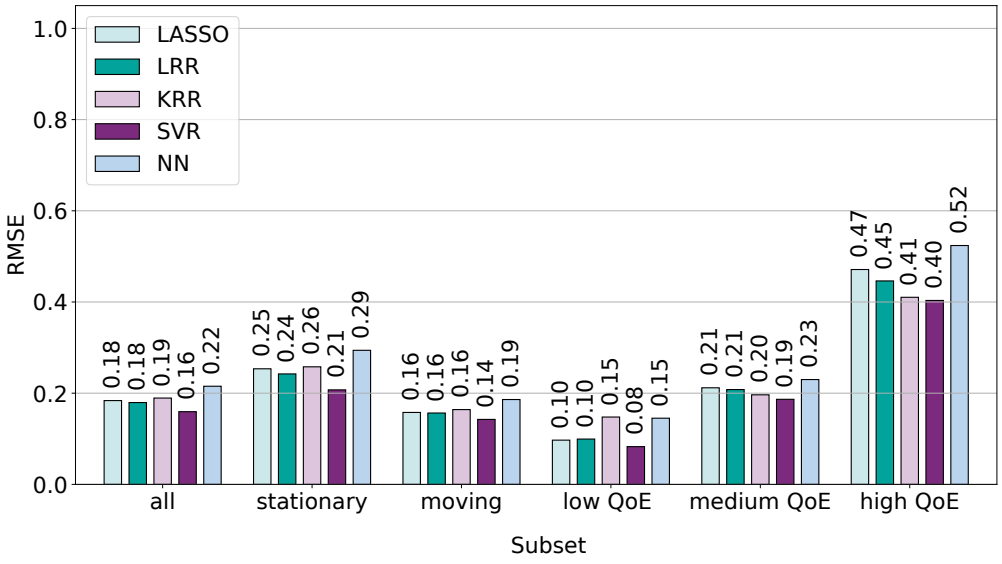


Figure 5.10: RMSE scores obtained for VoD-P.1203 within the different subsets.

magnitude of deviations from the true QoE score. The data points lie more far away from the angle bisector, indicating that inaccurate estimates are of higher magnitude with CQM.

For VoIP, the QoE estimates tend to be more accurate, as shown in Figure 5.9c. With LASSO, LRR, KRR, and SVR, the fractions of both, over- and under-estimation, are below 10%. SVR outperforms the other mechanisms and can achieve an accurate estimation within the +/-0.1-boundary for about 86.7% of the test samples. NN has a clear tendency towards under-estimating the QoE (29%), but over-estimates with a similar rate as the other four regression techniques.

5.5.1.2 Estimation Accuracy - RMSE

Next, we compare the five regression techniques using the root mean square error (RMSE). Thereby, we do not only consider the used application type, but also evaluate the estimation accuracy for different subsets of our ground-truth data set. For instance, when reporting the performance metrics, we take into account whether a UE was *stationary* or *moving*, and whether the ground-truth QoE is *low*, *medium*, or *high*. Please note that the models have not been explicitly trained on the subsets. Instead, they have been trained and optimized on the data set *all* and we only use the subsets when evaluating their performance.

The RMSE scores for VoD-P.1203 are illustrated in Figure 5.10. There are only slight differences between the five mechanisms in terms of RMSE in the complete data set (*all*). LASSO and LRR both obtain an RMSE of 0.18 and KRR yields 0.19. SVR is most accurate with an RMSE of 0.16, the NN has the highest error with 0.22. If the RMSE is considered separately for stationary and for moving clients, it shows that for VoD - P.1203, in any case, the estimation is more accurate for moving clients. This is most significant for SVR and

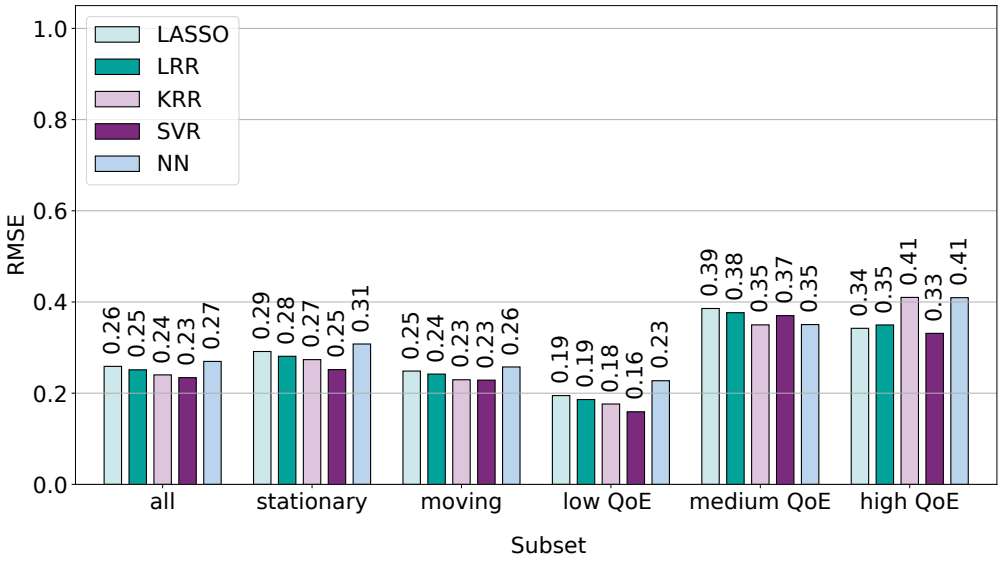


Figure 5.11: RMSE scores obtained for VoD-CQM within the different subsets.

NN, which yield an RMSE of 0.26 and 0.29 for stationary clients, but can achieve 0.16 and 0.19 for moving ones.

Next, we consider the RMSE separately depending on the true ground-truth QoE class. For low QoE scores, all techniques achieve a high estimation accuracy. SVR scores best with an RMSE of 0.08. The highest errors are obtained for NN and KRR with an RMSE of 0.15 each. For medium ground-truth QoE scores, the estimation errors increase for all five regression techniques. As before, SVR outperforms the other approaches and achieves an RMSE of 0.19. For high ground-truth QoE scores, we see again a decrease of the estimation accuracy. Compared to the low QoE subset, the RMSE scores are about the fourth-fold.

For the VoD-CQM (Figure 5.11), the estimation error in the overall data set is higher compared to VoD-P.1203. SVR achieves the lowest RMSE with 0.23 (0.16 for P.1203) and NN has the highest RMSE with 0.27. Again, the accuracy for moving clients is slightly better. However, the difference is less significant as with the P.1203 samples. Furthermore, low QoE scores can be estimated with higher accuracy than medium or high ones. The highest RMSE in the *low QoE* subset is 0.23 obtained from NN. This is still lower than the lowest RMSE in the *medium QoE* subset, which is 0.35 for KRR.

Finally, we investigate the estimation accuracy for VoIP, as shown in Figure 5.12. Considering the whole dat aset (*all*), SVR achieves the lowest RMSE with 0.10 and LASSO the highest with 0.17. In general, the estimation is more accurate for VoIP, compared to VoD. Contrary to VoD, the QoE of stationary clients can be estimated with higher accuracy than the QoE of moving clients. As we showed in Table 5.4, moving VoIP clients have lower MOS standard deviation than the stationary ones. For VoD, this is vice versa and the moving clients have a lower MOS standard deviation. Furthermore, Figure 5.12 shows that the VoIP QoE estimation is more accurate for higher ground-truth QoE values. While KRR achieves the best estimation accuracy in the *low QoE* subset with an RMSE of 0.48, it

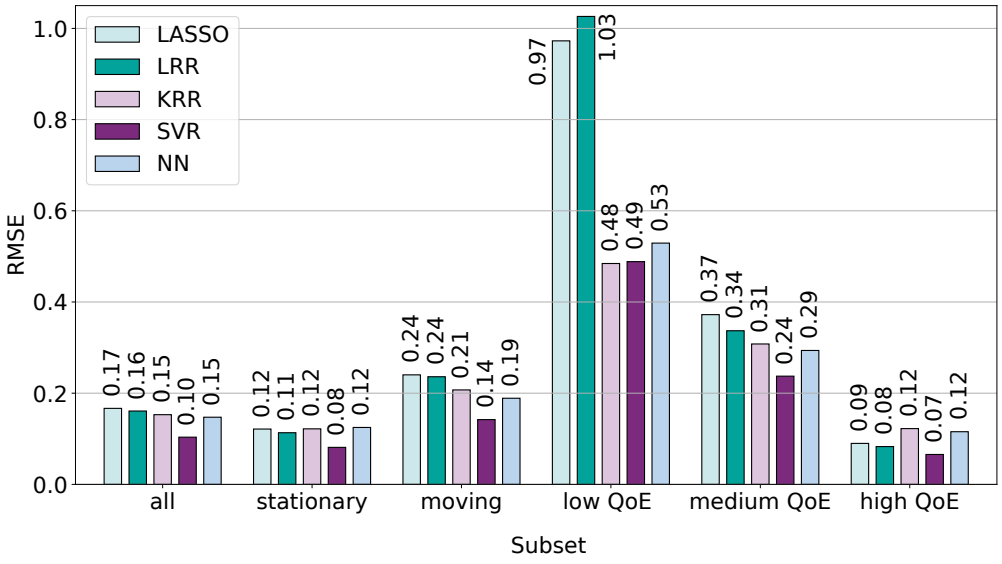


Figure 5.12: RMSE scores obtained for VoIP within the different subsets.

yields the worst performance in the *high QoE* subset, but with a significantly lower RMSE of only 0.12. This can be explained by the lower amount of ground-truth samples in the *low QoE* subset. Additionally, we showed in Figure 5.7c that with lower QoE scores, the values of the feature *std RLC Delay DL* are more spread along the x-axis, i.e., the relationship between the feature and QoE becomes less distinct and consequently makes the estimation of those samples more difficult. Finally, we note the low estimation accuracy of LASSO and LRR in the *low QoE* subset. While the other three mechanism yield an RMSE around 0.5, it is 0.97 for LASSO and 1.03 for LRR. A possible explanation for the low performance of these two linear models is their limited capability to capture the non-linear relationship between delay variation (jitter) and QoE accurately. Although the non-linear models also outperform the linear ones in the data set *all*, the effect is more obvious when it comes to the low QoE scores, where the delay and its variation actually play a role.

5.5.1.3 Estimation Accuracy - Further Metrics

Apart from the RMSE, we also compare the different mechanisms based on the MSE, the MedAE, and based on the correlation between estimated QoE and true QoE score according to PCC and SROCC. The respective scores are denoted in Table 5.5. For PCC and SROCC, higher values reflect a better performance (↑), while for MSE and MedAE, values as low as possible are desired (↓).

For the PCC, SVR is the mechanism which most often outperforms the other mechanisms. There are only few cases where SVR is significantly worse in terms of PCC compared one other approach. One is in the subset of low QoE scores for VoD-P.1203, where LASSO achieves a PCC of 0.76, while SVR scores 0.66. Another case is the subset of high QoE scores for VoD-CQM. The NN outperforms SVR with an increase of PCC by 0.05. Finally,

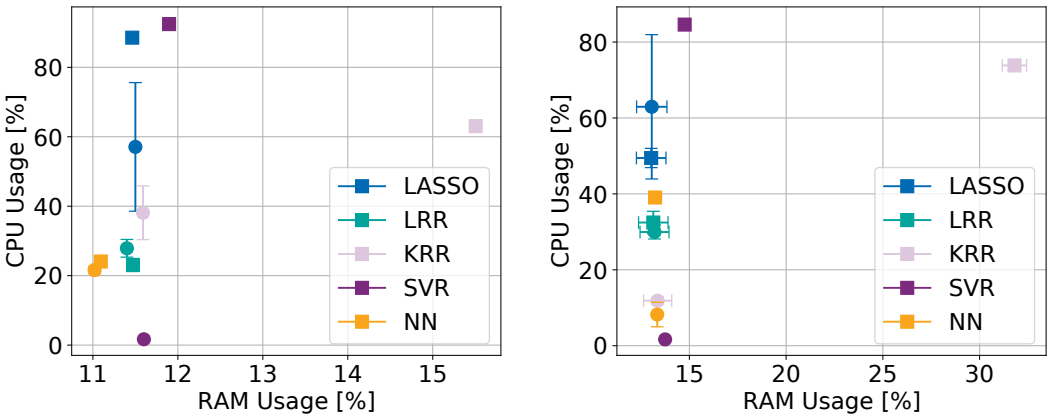
Table 5.5: Performance scores of the mechanisms within the different subsets. For PCC and SRCOCC, high values reflect a better score (↑), for MSE and MedAE, values should be as low as possible (↓). The respective best scores are highlighted as bold numbers.

		PCC (↑)					SROCC (↑)					MSE (↓)					MedAE (↓)				
		LASSO	LRR	KRR	SVR	NN	LASSO	LRR	KRR	SVR	NN	LASSO	LRR	KRR	SVR	NN	LASSO	LRR	KRR	SVR	NN
VoD - P.1203	all	0.95	0.95	0.95	**0.96**	0.94	0.93	0.93	**0.94**	0.92	0.86	0.034	0.032	0.036	**0.025**	0.046	0.075	0.075	**0.047**	0.059	0.122
	stationary	0.95	0.96	0.95	**0.97**	0.94	0.95	0.95	0.94	0.92	0.90	0.064	0.059	0.066	**0.043**	0.086	0.122	0.112	0.072	**0.077**	0.147
	moving	0.94	0.94	0.94	**0.95**	0.93	0.92	0.91	**0.94**	0.93	0.85	0.025	0.024	0.027	**0.020**	0.035	0.069	0.068	**0.044**	0.055	0.117
	low QoE	**0.76**	0.74	0.52	0.66	0.49	0.73	0.71	**0.78**	0.69	0.52	0.009	0.010	0.022	**0.007**	0.021	0.054	0.054	**0.028**	0.041	0.114
	medium QoE	0.83	0.83	0.86	**0.87**	0.81	0.83	0.84	0.87	**0.88**	0.81	0.045	0.043	0.039	**0.035**	0.053	0.131	0.126	0.109	**0.106**	0.132
	high QoE	0.66	**0.67**	0.64	0.66	0.59	0.66	0.66	0.64	0.66	0.56	0.222	0.199	0.168	**0.163**	0.274	0.311	**0.287**	0.293	0.309	0.410
VoD - CQM	all	0.94	0.95	0.95	0.95	0.94	0.95	0.94	**0.96**	0.95	0.91	0.067	0.063	0.058	**0.055**	0.073	0.134	0.121	**0.079**	0.081	0.087
	stationary	0.96	0.96	0.96	**0.97**	0.95	0.97	0.97	0.96	0.95	0.94	0.085	0.079	0.075	**0.063**	0.095	0.173	0.165	0.105	0.110	**0.104**
	moving	0.93	0.94	0.94	0.94	0.93	0.94	0.93	**0.95**	0.94	0.90	0.062	0.059	0.053	**0.052**	0.066	0.123	0.112	**0.073**	0.074	0.087
	low QoE	0.89	0.89	0.89	0.89	0.83	0.88	0.87	**0.91**	0.88	0.79	0.038	0.035	0.031	**0.025**	0.052	0.105	0.095	**0.050**	0.059	0.087
	medium QoE	0.69	0.68	**0.70**	0.69	0.66	0.67	0.66	**0.68**	0.67	0.64	0.149	0.142	**0.122**	0.137	0.123	0.288	0.279	**0.230**	0.256	0.243
	high QoE	0.61	0.66	0.51	0.62	**0.67**	0.66	**0.70**	0.53	0.66	0.69	0.117	0.122	0.168	**0.110**	0.168	**0.203**	0.256	0.249	0.209	0.310
VoIP	all	0.94	0.94	0.96	**0.98**	0.96	0.71	0.71	0.75	0.75	0.63	0.028	0.026	0.023	**0.011**	0.022	0.026	0.023	**0.011**	0.032	0.082
	stationary	0.94	0.95	0.94	**0.98**	0.96	0.66	0.66	0.71	**0.72**	0.56	0.015	0.013	0.015	**0.007**	0.016	0.024	0.022	**0.010**	0.030	0.082
	moving	0.93	0.94	0.96	**0.98**	0.96	0.79	0.80	0.82	0.82	0.75	0.058	0.056	0.043	**0.020**	0.036	0.032	0.027	**0.014**	0.037	0.081
	low QoE	0.55	0.49	**0.75**	0.49	0.10	0.82	**0.84**	0.80	0.69	0.24	0.946	1.053	**0.235**	0.239	0.280	0.389	0.291	0.259	**0.177**	0.353
	medium QoE	0.72	0.76	0.80	**0.87**	0.83	0.69	0.72	0.82	**0.85**	0.80	0.139	0.114	0.095	**0.056**	0.086	0.252	0.222	0.152	**0.143**	0.180
	high QoE	0.90	0.91	0.84	**0.94**	0.90	0.62	0.61	0.67	**0.68**	0.51	0.008	0.007	0.015	**0.004**	0.013	0.024	0.021	**0.009**	0.029	0.079

for the low VoIP QoE subset, KRR achieves a PCC of 0.75 while SVR only yields a PCC of 0.49. The subset of low VoIP QoE is also interesting for the NN, which only has a correlation of 0.10, the lowest value observed during our evaluations. According to SROCC, the mechanism which most frequently yields the highest values is KRR. Significantly higher SROCC are only achieved with LRR and NN in the subset of high VoD-CQM QoE. Similar as for PCC, the Neural Network fails in the low VoIP QoE dat aset, with a SROCC score of only 0.24. In terms of MSE, SVR outperforms any other mechanism for all subsets of VoD-P.1203. SVR also performs well on the VoD-CQM data set. Only for the medium VoD-CQM QoE subset, KRR and NN achieve a slightly better accuracy. In the VoIP data set, SVR's performance can only be topped in the low QoE subset by KRR. However, the difference is only 0.004 therefore negligible. Next, we investigate the performance based on the MedAE For this metric, KRR yields the lowest errors most frequently. Significantly lower errors can only be achieved by SVR in the stationary clients subset of VoD-P.1203, by LASSO and SVR in the high QoE subset of VoD-CQM, and by SVR for low and medium VoIP QoE.

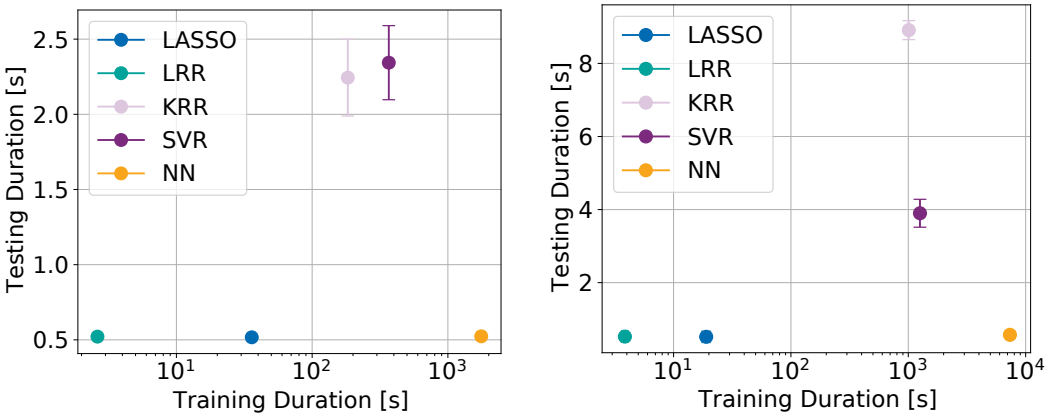
5.5.1.4 Meta-KPI Analysis

Our meta-analysis for the performance of the different regression techniques includes several metrics expressing the computational overhead of the ML models. Figure 5.13 denotes the CPU and RAM usage during training and testing, i.e., actually estimating the QoE for the samples in the testset. The metrics obtained for VoD are shown in Figure 5.13a. LRR and NN are CPU- and RAM-efficient during training. KRR has a higher RAM usage compared to the other mechanisms, while SVR and LASSO have the highest CPU-load during training. During testing, the most CPU-efficient approach is SVR and the most-RAM efficient approach is NN. Figure 5.13b shows the resource consumption for VoIP. Again, KRR has the highest RAM usage during training, however, in another order of magnitude. While with VoD, it used 15.5% of the RAM, it increases to 31.8% with VoIP. The RAM usage is similar for the remaining approaches and they mainly differ in terms of their CPU usage.



(a) VoD. **(b)** VoIP.

Figure 5.13: Resource consumption of the different regression techniques. Squares denote values obtained for training, circles for testing, respectively. Errorbars denote the standard deviation obtained after five repetitions.



(a) VoD. **(b)** VoIP.

Figure 5.14: Duration for training and testing the models. Errorbars denote the standard deviation after five repetitions.

The highest CPU usage during training is observed for SVR, the lowest one for LRR. During testing, similar as with VoD, SVR is the most CPU-efficient approach and LASSO the least efficient one.

Finally, we denote the duration for training and testing in Figure 5.14. Please note that the training duration, denoted on the x-axis, highly depends on the number of parameter combinations used to train the respective models and that a different amount of combinations has been used for the different approaches. Consequently, we will focus on the duration for testing, i.e., actually estimating the QoE. The results for VoD are shown in Figure 5.14a. LRR, LASSO, and NN are capable to estimate the QoE of the 2790 test samples in about 0.52 seconds. KRR and SVR, the methods applying the kernel trick, are less efficient and need about 2.3 seconds. Figure 5.14b shows the respective results for VoIP. We point out

Table 5.6: Classification of the evaluated mechanisms with respect to different qualitative aspects.

Topic	*Description*	LASSO	LRR	KRR	SVR	NN
Complexity	Comprehensibility by humans	Easy	Easy	Medium	Medium	Hard
	Number of settable hyper-parameters	1 (λ)	1 (λ)	2 (λ and kernel) + kernel-specific settings	3 (C, ε, kernel) + kernel-specific settings	~ 10
	Number of model parameters to tune	max #features+1	#features+1	#samples	#samples	#NN connections
	Complexity to finding the (near) optimum	Easy	Easy	Medium	Medium	Hard
Data set	Requirements on the data set size	Low	Low	Medium	Medium	High
	Sensitivity towards outliers	High	High	Medium	Low	Medium
Feature selection	Detection of relevant features	Easy (Integrated)	Dedicated step needed (e.g. PCA, correlation analysis)			
	Reducing the number of used features	Easy (tuning λ)	Hard/Dedicated step needed if not all features should be used			
Trackability	Feasibility to track a model's evolution	Easy	Medium	Medium	Medium	Hard
	Amount of data that needs to be tracked	Low	Low	Low	Low	Medium
Over-fitting	Sensitivity towards over- or under-fitting	Low	Low	Medium	Medium	Medium
	Automatically applied prevention mechanisms	Yes	Yes	Yes	Yes	No
	Efforts to prevent over-fitting	Low	Low	Low	Low	Medium
Service coverage	Applicability to different types of problems	Low	Low	High	High	High

that the VoIP testset contains 5518 test samples, roughly the double compared to the VoD testset. Again, LRR, LASSO, and NN are the fastest estimators and need about half a second. It takes 3.9 seconds for SVR, and KRR has the longest estimation duration with about 8 seconds.

5.5.2 Qualitative Assessment

Besides the typical performance metrics, stakeholders such as MNOs also need to take qualitative aspect into account, when deciding which algorithm to deploy in their networks. In the following, we use qualitative scales and classify the algorithms applied in this work with respect to different design decisions, as shown in Table 5.6.

5.5.2.1 Requirements on Data Set Size

Primarily, the required size of the data set, which allows to train the model adequately, depends on the complexity of the problem. Nevertheless, the required amount of data also depends on the used model itself. In general, more complex models (i.e., with a high dimensional weight parameter space) require more data, unless unreasonably strong regularization is applied. The linear models, LASSO and LRR, have relatively low dimensional parameter space (usually the number of features plus one), and hence their requirements in terms of the number of samples in the training data set are comparably low. On the other hand, KRR and SVR with non-linear kernels typically require more data in order to learn the higher order dependencies. The complexity of NN depends on the architecture, but its benefits appear when a complex architecture is trained on a large data set. In this sense, it requires a large amount of data. Outliers are critical in general when it comes to training ML algorithms and should be eliminated during the pre-processing step, if possible. The sensitivity towards outliers mainly depends on the loss function to be minimized — squared losses, e.g., MSE,

are highly sensitive to outliers, while (piece-wise) linear loss is robust. Therefore, SVR trained with the piece-wise linear ε-insensitive loss tends to be more robust than the other four models.

5.5.2.2 Training and Hyper-parameter Tuning

For the ML model for which the training is performed by solving a convex problem, reliable solvers are accessible — the problem is solved analytically or by an iterative algorithm with convergence guarantee. The training objectives for LASSO, LRR, KRR, and SVR are all convex, and therefore, we can expect that the model is stably trained when the model hyper-parameters are set appropriately. On the other hand, NN is trained by solving non-convex problems, and available state-of-the-art solvers are only guaranteed to converge to a local solution. There are many tips on choice of solvers, e.g., stochastic gradient descent or ADAM [161], initialization, and algorithm parameter setting, e.g., learning rate, momentum, the number of epochs, batch size, to likely get a "good" local solution, but good setting can depend on the model architecture and the model hyper-parameters, and therefore, cannot be fully automatic and human intervention is necessary when training fails. The model hyper-parameters have to be set appropriately. The linear methods, LASSO and LRR, have a single regularization parameters, which can be tuned by grid search on the validation error, i.e., prediction error on validation data. The kernel methods, KRR and SVR, have bandwidth parameters, which should be tuned, although the default value ($\gamma = 1$) can also work, assuming that the training data is appropriately pre-processed, e.g., standardization so that all features have zero means and unit variances. For NN, the architecture corresponds to the model hyper-parameters, including the number of hidden layers, the neuron type, e.g., fully-connected, convolutional, pooling, etc., and the number of nodes in each layer. For extensive exploration, Bayesian optimization can be used.

5.5.2.3 Feature Selection and Interpretability

Finding the features that are relevant for estimating the QoE is important. Practically, this information can be used to reduce the costs for estimation (e.g., collecting the data and processing it to generate the features), and allows to understand what the ML model has learned, or explain why it predicts a particular response for particular input features. The latter is important when ML models are deployed in real applications that require high reliability and security. For linear models, the learned weights can be seen as the importance of the corresponding features, and therefore are easily interpretable – the features for which the learned absolute weights are large are relevant for predicting the response. However, correlations between features can cause spuriously detect relevant features, because the contribution from the two positively correlated features with large positive and negative weights, respectively, can cancel to each other. LASSO was proposed to avoid this phenomenon. The sparsity inducing L1 regularizer suppresses the contribution (i.e., weight) from unnecessary features, and its solution for a fixed number of non-zero weights

are guaranteed to correspond to the set of features that best predicts the response. The kernel methods and NN are seen as black-box predictors, and there is no straightforward way to interpret. In a recently emerging research field, called explainable artificial intelligence (XAI) [162], researchers are tackling to address this issue, and many methods have been and are being developed. However, no existing method is guaranteed to correctly explain the ML model, and furthermore, vulnerability against adversarial attack was pointed out [163]. Accordingly, interpreting non-linear ML models is so far a relatively hard task, requiring at least some human effort and expert knowledge.

5.5.2.4 Trackability

If an ML model is re-trained regularly using new ground-truth data, it will evolve over time. This happens for diverse reasons. The content provider could change the model used to compute the QoE which is communicated via the AF or it could adjust application settings. For the example of VoD, this could be a change in the quality switching thresholds, a reconfiguration of the maximum amount of buffered playtime, or changing video encoding characteristics, e.g., the segment duration or video bitrate. In case of VoIP, such a change could be the implementation of a new voice codec. Furthermore, changes regarding the network configuration, such as applying another scheduling algorithm for resource allocation at the AN, influence the correlation between network-related features and QoE. It might be of interest for an MNO, to monitor how the model evolves over time. For instance, to track which features gained importance and which ones became less relevant. Accordingly, the trackability of a model mainly depends on three factors, which have previously been discussed: Its capability for feature relevance analysis, its comprehensibility, and its number of (hyper-)parameters. Tracking how the model evolves over time is very simple with LASSO. It returns a p-value for each feature, which can be seen as a measure of its respective importance to estimate QoE. Besides, λ is the only configurable parameter that needs to tracked. With LRR, KRR, and SVR, such a simple tracking of feature importance cannot be performed. Indeed, the weights of the input features could be seen as a rough approximation of their importance, but this requires a linear kernel and that all features have the same scale, which is seldom the case. Hence, if the feature importance should be tracked, dedicated methods need to be applied, also beyond the training process. This allows to keep track of the feature importance in general, but not the feature importance with respect to the specific model which was applied. However, only few model parameter settings need to be monitored and we classify the trackability of KRR, LRR, and SVR as medium. To obtain the feature importance with NN, more complex methods, like permutation importance, need to be applied. The values of the features are, one after another, randomly shuffled. These shuffled values are used as an input for the trained model. Analyzing how much the prediction output is distorted by modifying the input, allows to estimate the importance of a features. As this process is very inefficient, and a huge range of model parameters need to be tracked, we classify the NN as hard in terms of trackability.

5.5.2.5 Service Coverage

The range of problems to which a specific ML algorithm can be applied is varying. In the context of estimating QoE, we refer to this range as the service coverage, i.e., to how many service types an algorithm can be applied without knowing the relationship between input features and QoE upfront. Due to their linearity, LASSO and LRR are limited to services where this relationship is linear. Contrary, KRR, SVR, and NN can be used for most problems, even if the relationship between input features and response is not linear. Thereby, the NN with its deep architecture might show advantages in solving highly complicated problems.

5.6 Lessons Learned

In this chapter, we elaborated on the capabilities of new NFs introduced with 5G and how they can be exploited to overcome current QoE monitoring limitations. More specifically, we proposed an approach relying on the AF and the NWDAF, allowing an MNO to utilize ML so as to estimate the QoE solely based on network KPIs, which it can access at any time. In this context, we discussed the involved challenges and design criteria from an MNO's point of view and conducted a feasibility study to demonstrate the applicability of the proposed solution.

From our study, we learned that it is possible to obtain a reliable QoE estimation from network telemetry data. More specifically, we trained and evaluated a set of five distinct regression techniques, which are representative in terms of the their complexities, and studied their accuracy with respect to different performance metrics. In order to draw more generalizable conclusions, we used with VoD and VoIP two distinct types of applications, which differ with respect to their QoS/QoE relationship. To be able to identify possible influence factors on the achieved performance, our study considers heterogeneous movement patterns of clients and examines the accuracy within different subsets, representing, e.g. low, medium, or high QoE scores.

The lowest estimation accuracy along the complete ground-truth data set is an RMSE of 0.27, obtained for the CQM model using the NN, which can still be seen as a very accurate estimation. However, we learned that linear models, i.e., LASSO and LRR, fail to estimate the *low QoE* scores for VoIP. This is due to the non-linear relationship between large delays, i.e., a highly important KPI for VoIP degradation, and the resulting QoE. Consequently, when assessing the applicability of different ML techniques with respect to estimating the QoE of a given service, either the complex relationships between QoS and QoE need be known beforehand or thoroughly examined. Otherwise, several options for a possible deployment should be considered, allowing to choose the best performing one.

The estimation performance is, however, not the only factor determining the suitability of a specific ML model. Instead, there are several practical consideration which should be

taken into account, as discussed during the qualitative comparison of the different regression techniques. We found that while the non-linear approaches, especially KRR and SVR, performed well in all of the investigated scenarios and subsets, their decisions are hard to trace, and hence, could impede a root cause analysis. Finally, we note that the ML option to be deployed should be well tailored to the underlying use-case, as we identified a high heterogeneity of the different models in terms of their CPU-load, RAM-usage, and their duration for training and testing, despite using the same physical machine and frameworks.

6

Conclusion

Novel applications running on today's Internet, such as UHD video streaming or VR gaming, have ever increasing resource demands and are thus challenging the underlying network. In addition to that is the number of users of such services steadily growing, as well as the users' expectations on being provided with a good quality. The paramount goal of network and service providers is to keep the users satisfied, whilst operating in an economical manner. Otherwise, they cannot be competitive in the constantly expanding market.

While the technology-oriented concept of QoS management allows for controlling and managing the network resources, it neglects the end-user along the service chain. QoE management overcomes this limitation by additionally considering end-user related factors such as the used application, the delivered content, or the context. Over the past years, QoE management has emerged as an important research area, due to its huge potential of solving the formerly mentioned challenges. For instance, it involves mechanisms for efficiently exploiting network resources and for applying advanced application-level mechanisms so to achieve customer loyalty by delivering a good QoE, whilst at the same time being economically viable by means of profitable service operation.

Despite the vast research efforts related to QoE management, it still faces a multitude of challenges to date. To exploit its full potential, it is essential to co-ordinate the control loops on application- and network-level. Thereby, the key limitation is that NPs and APs have distinct angles on the Internet ecosystem. Consequently, each of them operates within their specific action space and hence – to a certain degree – independent of the respective other. For instance, the AP has capabilities to assess the QoE or its relevant parameters,

as well as to collect in-app user ratings. However, it is either unaware of the underlying network KPIs or has to rely on rough estimations of the current network state when tuning application-specific settings. Analogous to that has the NP no information about detailed application specifics, whereas it is capable of performing fine-grained network monitoring as well as to implement advanced resource allocation strategies.

This monograph covered three different aspects related to QoE management, with the goal to overcome its current limitations and to improve the user satisfaction on the example of HAS. Thereby, the focus was set on all three QoE management building blocks, i.e., QoE modeling, QoE optimization, and QoE monitoring, and the lessons learned have relevance for both of the main stakeholder in the HAS ecosystem, the AP as well as the NP. More specifically, we could derive the following conclusions:

Analytical models allow to understand the complex interplay between application- and network-specific parameters and their impact on QoE. In the course of this monograph we extended an existing analytical approach, relying on discrete-time analysis, by the capability of modeling the HAS-specific quality adaptation feature, according to both, a buffer-based and a rate-based adaptation scheme. The conducted validation confirms the model's reliability in terms of computing the buffer state and other important KPIs, allowing for efficiently retrieving a well tuned parameter setting, so as to optimize the QoE delivered to the user. We furthermore showed that, despite the probabilistic nature of the model's output and the resulting lack of temporal context, it can still reflect the QoE as obtained with time-dependent QoE estimation models, such as P.1203. To demonstrate the model's applicability for optimization purposes, we performed an exemplary parameter study from which we concluded that the threshold for leaving the lowest quality level should rather be set in a conservative manner. Compared to an aggressive setting, it reduces stallings in scenarios with moderate to high bandwidth fluctuations, but is still capable of achieving the same average video quality in scenarios with low or no bandwidth variation. From the presented efficient way to build up HAS-specific domain knowledge to support QoE modeling activities, we exemplarily derived a specific configuration which can lead to a general QoE improvement for HAS, i.e., independent of the network characteristics.

Variable segment durations can enhance the efficiency of the HAS ecosystem by reducing the video encoding overhead and improving the delivered QoE. Typically, videos for HAS are split in a content-agnostic manner in order to achieve video segments of equal durations. In the scope of this monograph we considered a segmentation technique which takes the video content into account by splitting at scene-cuts, resulting in segments of variable durations. This allows to significantly reduce the number of costly I-frames and thus, to increase the overall encoding efficiency of HAS-prepared videos. By means of our data set, comprising a vast number of video sequences encoded with a representative set of different settings, we derive the relevant factors which influence the magnitude of bitrate reduction achieved by the variable approach. Moreover, we conducted controlled testbed measurements, considering various bandwidth capacities and different ABR mechanisms, to study the impact of variable segment durations on the video streaming QoE. We

learn that in scenarios with low bandwidth limits, the variable approach reduces the number of video interruptions, delivers a higher visual quality, and hence, clearly outperforms the state-of-the art mechanism. However, with increasing bandwidth capacities, we observe more cases where the variable approach degrades the subjective HAS performance. Due to a faster buffer ramp-up, higher qualities are more likely to be requested. This behavior, combined with the enlarged segment size variability at higher quality levels, increases the risk of stallings. To overcome this issue, the quality switching thresholds could either be set more conservative, or dedicated ABR strategies should be applied, which are capable of taking the segments' sizes into account when selecting the quality. This small adaptation to the system would allow to fully exploit the potential of variable segment durations.

The introduction of new features for third party information exchange and data analytics in 5G can overcome current QoE monitoring limitations. The two key enabling 5G NFs discussed in this monograph are the NWDAF and the AF. The NWDAF is dedicated for collecting, processing, and analyzing a vast amount of data and it can share generated analytics with other NFs. The AF provides a standardized interface for the communication with third parties, such as APs. In the scope of this work, we proposed a work flow relying on these NFs, which are newly introduced with the 5th generation of mobile networks. More specifically, we assume that the MNO receives QoE information from the AP via the AF. The MNO is thus capable of correlating the obtained ground-truth QoE with network statistics available at the NWDAF in order to train ML-based models. During deployment, such a model is capable of estimating the QoE from network telemetry data, which can be collected by the MNO at any time. While this can potentially overcome current QoE monitoring limitations of NPs, it is associated with several challenges and design criteria. We first elaborated on the key challenges from the viewpoint of an MNO and then demonstrated by means of a simulation-based study the feasibility of such an approach. Therefore, we compared five regression techniques with respect to their estimation accuracy, resource consumption, and duration for training and testing. In addition to that, we performed a qualitative comparison of the techniques, taking into account aspects such as their comprehensibility or their capability of performing feature selection. Our results and discussions can serve as a guideline for ML deployment in 5GS and show in how far the tested techniques can satisfy the requirements and cope with the heterogeneity of today's networks.

In general, introducing more context information to the HAS ecosystem allows to run it in a more efficient manner. We have shown that by taking the video content into account, instead of purely relying on a content-agnostic segmentation, a more efficient video encoding can be achieved. In order to fully exploit the potential of variable segment durations for HAS, it is advisable to not only increase the amount of considered context for encoding, but also for streaming. For instance, in a next step, one could contemplate to dynamically tune quality switching and buffer thresholds to the video segments' characteristics, such as their average segment size and variation, instead of relying on a one-fits-all solution. Although this additionally increases the complexity of the system, it will eliminate the drawbacks of the variable approach identified in this monograph. Such optimized threshold settings could, for example, be revealed be applying the proposed analytical queuing model.

Enhanced context-awareness also plays an essential role for 5G and beyond networks, among others driven by the goals of improving network efficiency and providing a better QoE. The 5G architecture also introduces new mechanisms for resource allocation, such as QoS-flows, which allow for assigning per-flow bitrate guarantees, or network slicing, a technology to run dedicated virtual networks tailored to a vertical's specific needs, e.g. ultra low delay or high bandwidth, on the same physical infrastructure. Accordingly, the next steps should focus on the exploitation of such new 5G-specific control mechanisms in an optimized and QoE-aware manner. For instance, the obtained QoE information can be used to enhance the user satisfaction by triggering a well targeted, automated resource allocation. The implementation of an AI-enabled, self-organized control loop, consisting of QoE monitoring, deriving control actions, and reviewing their implications on the QoE, can form the basis for fully autonomous QoE management in next generation networks.

List of Figures

List of Tables

Glossary

NWDAF Network Data Analytics Function. 90, 98, 99, 101–104, 122, 126

QoE-IFs QoE Influence Factors. 10, 17, 20, 43, 52, 53, 56

3G Third Generation of Mobile Telecommunications Technology. 80

4G Fourth Generation of Mobile Telecommunications Technology. 97, 104

5G Fifth Generation of Mobile Telecommunications Technology. ii, 3, 5, 7, 9, 90, 97–99, 101, 103, 104, 122, 126, 127

5GS 5G Systems. 89, 90, 98, 126

ABR Adaptive Bitrate Algorithm. 10–13, 18–20, 28, 57, 58, 63, 65, 79, 81, 88, 125, 126

AF Application Function. 90, 99, 101–103, 122, 126

AI Artificial Intelligence. 101, 127

AMF Mobility Management Function. 101

AN Access Node. 103–105

AP Application Provider. 2–5, 89, 90, 99, 102, 103, 124–126

CBR Constant Bitrate. 58, 69–71, 73–76, 79

CQI Channel Quality Indicator. 110

CQM Cumulative Quality Model. 17, 104, 108–110, 112, 114, 122

CRF Constant Rate Factor. 54, 55, 58, 69–71, 73, 74, 77, 78, 87

DASH Dynamic Adaptive Streaming over HTTP. 80

ECDF Empirical Cumulative Distribution Function. 74, 76, 82, 108

FR Full Reference. 15, 60, 105

FVQA Fusion-based Video Quality Assessment. 15

GOP Group of Pictures. 60

HAS HTTP Adaptive Streaming. i, 4–7, 9–20, 27–31, 38, 41, 44, 47–52, 54–60, 62, 63, 65, 80, 87–89, 103, 104, 125, 126

HTTP Hypertext Transfer Protocol. 11, 13, 62

IP Internet Protocol. i, 4, 42, 87

ISP Internet Service Provider. 97

KPI Key Performance Indicator. ii, 3, 4, 7, 9, 10, 90, 97, 98, 100, 102, 103, 105, 112, 122, 125

KRR Kernel Ridge Regression. 94, 103, 107, 112, 114, 115, 117–123

LASSO Least Absolute Shrinkage Operator. 92, 93, 103, 107, 112, 114–122

LRR Linear Ridge Regression. 93, 94, 98, 103, 107, 114, 116–122

LTE Long Term Evolution. 97

MedAE Median Absolute Error. 96, 116, 117

ML Machine Learning. ii, 7, 9, 90, 91, 96–101, 103, 105–109, 117, 122, 123, 126

MNO Mobile Network Operator. ii, 7, 9, 90, 97–103, 105, 109, 122, 126

MOS Mean Opinion Score. 15, 17, 49, 50, 77, 82, 83, 88, 95, 97, 103, 105, 108–110, 112, 115

MSE Mean Squared Error. 94, 96, 107, 116, 117

NF Network Function. ii, 5, 89, 90, 98–103, 122, 126

NMS Network Management System. 101

NN Neural Network. 95, 96, 98, 103, 107, 112, 114–122

NP Network Provider. 2–5, 89, 124–126

NR No Reference. 15

OTT Over the Top. 97

PCC Pearson's Correlation Coefficient. 96, 110, 111, 116, 117

PGW Packet Gateway. 104

POI Point of Interest. 104, 105

PSNR Peak Signal-to-Noise Ratio. 15, 62, 71

QoE Quality of Experience. i, ii, 1–7, 9–11, 13–19, 27, 46–52, 54–58, 78, 80–86, 88–90, 92, 93, 95, 97–105, 107–110, 112–118, 120–122, 124–127, 129, 130

QoS Quality of Service. 2–4, 10, 89, 97

QP Quantization Parameter. 81

RLC Radio Link Control. 105, 110

RMSE Root Mean Squared Error. 96, 114–116, 122

RR Reduced Reference. 15

RTT Round Trip Time. 17, 45, 62, 106

RV Random Variable. 22–24, 26, 45

SBI Service Based Interface. 98, 101, 102

SDN Software Defined Networking. 97

SI Spatial Information. 67, 68

SMF Session Management Function. 101

SROCC Spearman's Rank-order Correlations. 96, 116, 117

SSIM Structural Similarity. 15, 58–62, 71, 76–78

SVR Support Vector Machine. 94, 103, 107, 112, 114–123

TCP Transmission Control Protocol. 42, 49, 62, 106

TI Temporal Information. 67, 68

TLS Transport Layer Security. 97

UE User Equipment. 98, 103–105, 109, 110

UHD Ultra High Definition. 1, 124

UPF User Plane Function. 104

VBR Variable Bitrate. 58, 69–71, 73–79

VM Virtual Machine. 80

VMAF Video Multimethod Assessment Fusion. 15

VoD Video on Demand. 1, 10, 104, 105, 107–110, 112, 114, 115, 117–119, 121, 122

VoIP Voice over IP. 1, 3, 10, 97, 100, 103–105, 107–110, 114, 115, 117–119, 121, 122

VoLTE Voice over LTE. 97

VR Virtual Reality. i, 3, 124

Bibliography

[1] S. Schwarzmann, P. Breitbach, and T. Zinner. “Computing QoE-relevant adaptive video streaming metrics using discrete-time analysis”. In: *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE. 2019, pp. 1–6.

[2] S. Schwarzmann, P. Breitbach, T. Zinner, and M. Rost. “Modeling adaptive video streaming using discrete-time analysis”. In: *31st International Teletraffic Congress (ITC 31)*. IEEE. 2019, pp. 121–129.

[3] S. Schwarzmann and T. Zinner. “Linking QoE and performance models for DASH-based video streaming”. In: *6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2020, pp. 65–71.

[4] S. Schwarzmann, T. Zinner, S. Geissler, and C. Sieber. “Evaluation of the benefits of variable segment durations for adaptive streaming”. In: *Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2018, pp. 1–6.

[5] S. Schwarzmann, N. Hainke, T. Zinner, C. Sieber, W. Robitza, and A. Raake. “Comparing fixed and variable segment durations for adaptive video streaming: A holistic analysis”. In: *Proceedings of the 11th ACM Multimedia Systems Conference*. 2020, pp. 38–53.

[6] S. Schwarzmann, C. Cassales Marquezan, M. Bosk, H. Liu, R. Trivisonno, and T. Zinner. “Estimating video streaming QoE in the 5G architecture using machine learning”. In: *Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*. 2019, pp. 7–12.

[7] S. Schwarzmann, C. C. Marquezan, R. Trivisonno, S. Nakajima, and T. Zinner. “Accuracy vs. cost trade-off for machine learning based QoE estimation in 5G networks”. In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–6.

[8] S. Schwarzmann, C. C. Marquezan, R. Trivisonno, S. Nakajima, V. Barriac, and T. Zinner. “ML-based QoE estimation in 5G networks using different regression techniques”. In: *IEEE Transactions on Network and Service Management* (2022).

[9] C. Sieber, S. Schwarzmann, A. Blenk, T. Zinner, and W. Kellerer. "Scalable application- and user-aware resource allocation in enterprise networks using end-host pacing". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 5.3 (2020), pp. 1–41.

[10] M. Bosk, M. Gajic, S. Schwarzmann, S. Lange, R. Trivisonno, C. Marquezan, and T. Zinner. "Using 5G QoS mechanisms to achieve QoE-aware resource allocation". In: *17th International Conference on Network and Service Management (CNSM)*. IEEE. 2021, pp. 283–291.

[11] M. Bosk, M. Gajić, S. Schwarzmann, S. Lange, and T. Zinner. "HTBQueue: A hierarchical token bucket implementation for the OMNeT++/INET framework". In: *arXiv preprint arXiv:2109.12879* (2021).

[12] T. Zinner, F. Lemmerich, S. Schwarzmann, M. Hirth, P. Karg, and A. Hotho. "Text categorization for deriving the application quality in enterprises using ticketing systems". In: *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, Cham. 2015, pp. 325–336.

[13] S. Schwarzmann, T. Zinner, and M. Hirth. "Deriving the employee-perceived application quality in enterprise IT infrastructures using information from ticketing systems." In: *LWA*. 2014, pp. 69–70.

[14] S. Schwarzmann, T. Zinner, and O. Dobrijevic. "Quantitative comparison of application–network interaction: a case study of adaptive video streaming". In: *Quality and User Experience* 2.1 (2017), pp. 1–18.

[15] S. Schwarzmann, T. Zinner, and O. Dobrijevic. "Towards a framework for comparing application-network interaction mechanisms". In: *28th International Teletraffic Congress (ITC 28)*. Vol. 3. IEEE. 2016, pp. 13–18.

[16] B.-H. Chu, M.-S. Tsai, and C.-S. Ho. "Toward a hybrid data mining model for customer retention". In: *Knowledge-Based Systems* 20.8 (2007), pp. 703–718.

[17] Q.-T. Luu, S. Kerboeuf, and M. Kieffer. "Uncertainty-aware resource provisioning for network slicing". In: *IEEE Transactions on Network and Service Management (TNSM)* 18.1 (2021), pp. 79–93.

[18] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer. "Challenges of QoE management for cloud applications". In: *IEEE Communications Magazine* 50.4 (2012), pp. 28–36.

[19] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori. "QoE management of multimedia streaming services in future networks: a tutorial and survey". In: *IEEE Communications Surveys & Tutorials* 22.1 (2019), pp. 526–565.

[20] N. Barman and M. G. Martini. "QoE modeling for HTTP adaptive video streaming – A survey and open challenges". In: *IEEE Access* 7 (2019), pp. 30831–30859.

[21] W. Robitza, A. Ahmad, P. A. Kara, L. Atzori, M. G. Martini, A. Raake, and L. Sun. "Challenges of future multimedia QoE monitoring for internet service providers". In: *Multimedia Tools and Applications* 76.21 (2017), pp. 22243–22266.

[22] A. Ahmad, A. Floris, and L. Atzori. "QoE-aware service delivery: A joint-venture approach for content and network providers". In: *International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2016, pp. 1–6.

[23] V. Joseph and G. de Veciana. "NOVA: QoE-driven optimization of DASH-based video delivery in networks". In: *Conference on computer communications (INFOCOM)*. IEEE. 2014, pp. 82–90.

[24] R. Schatz, M. Fiedler, and L. Skorin-Kapov. "QoE-based network and application management". In: *Quality of experience*. Springer, 2014, pp. 411–426.

[25] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen. "A survey of emerging concepts and challenges for QoE management of multimedia services". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.2s (2018), pp. 1–29.

[26] A. Floris, A. Ahmad, and L. Atzori. "QoE-aware OTT-ISP collaboration in service management: Architecture and approaches". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.2 (2018), pp. 1–24.

[27] Cisco. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. White Paper. Cisco Syst., Inc., 2019.

[28] F. Agboma and A. Liotta. "QoE-aware QoS management". In: *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia (MoMM)*. ACM. 2008, pp. 111–116.

[29] N. Bhatti, A. Bouch, and A. Kuchinsky. "Integrating user-perceived quality into web server design". In: *Computer Networks* 33.1-6 (2000), pp. 1–16.

[30] S. Khirman and P. Henriksen. "Relationship between quality-of-service and quality-of-experience for public internet service". In: *Proceedings of the 3rd Workshop on Passive and Active Measurement (PAM)*. Vol. 1. 2002.

[31] P. Reichl. "From charging for Quality of Service to charging for Quality of Experience". In: *Annales des télécommunications*. Vol. 65. 3-4. 2010, pp. 189–199.

[32] M. Fiedler, K. Kilkki, and P. Reichl. "Executive summary – from quality of service to quality of experience". In: *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2009.

[33] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, et al. "Qualinet white paper on definitions of quality of experience". In: *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)* (2013).

[34] P. Juluri, V. Tamarapalli, and D. Medhi. "Measurement of quality of experience of video-on-demand services: A survey". In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 401–418.

[35] S. Möller, B. Naderi, C. Keimel, and D. Saupe. "Crowdsourcing Quality of Experience Experiments". In: *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments: Dagstuhl Seminar 15481, Revised Contributions*. Vol. 10264. Springer. 2017, p. 154.

[36] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. "Quantification of YouTube QoE via crowdsourcing". In: *International Symposium on Multimedia*. IEEE. 2011, pp. 494–499.

[37] K. Borchert, A. Seufert, E. Gamboa, M. Hirth, and T. Hoßfeld. "In vitro vs in vivo: does the study's interface design influence crowdsourced video QoE?" In: *Quality and User Experience* 6.1 (2020), pp. 1–16.

[38] H. J. Kim and S. G. Choi. "A study on a QoS/QoE correlation model for QoE evaluation on IPTV service". In: *The 12th International Conference on Advanced Communication Technology (ICACT)*. IEEE. 2010, pp. 1377–1382.

[39] M. Alreshoodi and J. Woods. "Survey on QoE/QoS correlation models for multimedia services". In: *arXiv preprint arXiv:1306.0221* (2013).

[40] H. J. Kim, D. H. Lee, J. M. Lee, K. H. Lee, W. Lyu, and S. G. Choi. "The QoE evaluation method through the QoS-QoE correlation model". In: *Fourth International Conference on Networked Computing and Advanced Information Management*. Vol. 2. IEEE. 2008, pp. 719–725.

[41] S. Moteau, F. Guillemin, and T. Houdoin. "Correlation between QoS and QoE for HTTP YouTube content in Orange cellular networks". In: *9th Latin-American Conference on Communications (LATINCOM)*. IEEE. 2017, pp. 1–6.

[42] Z. Wang, L. Li, W. Wang, Z. Wan, Y. Fang, and C. Cai. "A study on QoS/QoE correlation model in wireless-network". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE. 2014, pp. 1–6.

[43] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hoßfeld. "Impact of frame rate and resolution on objective QoE metrics". In: *Second International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE. 2010, pp. 29–34.

[44] T. Hoßfeld, P. E. Heegaard, L. Skorin-Kapov, and M. Varela. "No silver bullet: QoE metrics, QoE fairness, and user diversity in the context of QoE management". In: *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2017, pp. 1–6.

[45] *MPEG-DASH*. `https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash`. Accessed: 2021-11-07.

[46] T. Stockhammer. "Dynamic adaptive streaming over HTTP – Standards and design principles". In: *Proceedings of the ACM Conference on Multimedia Systems (MMSys)*. 2011, pp. 133–144.

[47] J. Kua, G. Armitage, and P. Branch. "A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP". In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1842–1866.

[48] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann. "A survey on bitrate adaptation schemes for streaming media over HTTP". In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 562–585.

[49] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo. "Buffer-based smooth rate adaptation for dynamic HTTP streaming". In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE. 2013, pp. 1–9.

[50] H. T. Le, D. V. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang. "Buffer-based bitrate adaptation for adaptive HTTP streaming". In: *International Conference on Advanced Technologies for Communications (ATC)*. IEEE. 2013, pp. 33–38.

[51] C. Mueller, S. Lederer, R. Grandl, and C. Timmerer. "Oscillation compensating dynamic adaptive streaming over HTTP". In: *International Conference on Multimedia and Expo (ICME)*. IEEE. 2015, pp. 1–6.

[52] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. "BOLA: Near-optimal bitrate adaptation for online videos". In: *IEEE/ACM Transactions on Networking* 28.4 (2020), pp. 1698–1711.

[53] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. "Probe and adapt: Rate adaptation for HTTP video streaming at scale". In: *IEEE Journal on Selected Areas in Communications* 32.4 (2014), pp. 719–733.

[54] C. Liu, I. Bouazizi, and M. Gabbouj. "Rate adaptation for adaptive HTTP streaming". In: *Proceedings of the second Conference on Multimedia systems (MMSys)*. 2011, pp. 169–174.

[55] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj. "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network". In: *Signal Processing: Image Communication* 27.4 (2012), pp. 288–311.

[56] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. "A control-theoretic approach for dynamic adaptive video streaming over HTTP". In: *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM. 2015, pp. 325–338.

[57] A. Sobhani, A. Yassine, and S. Shirmohammadi. "A video bitrate adaptation and prediction mechanism for HTTP adaptive streaming". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 13.2 (2017), pp. 1–25.

[58] G. Van Wallendael, W. Van Lancker, J. De Cock, P. Lambert, J.-F. Macq, and R. Van de Walle. "Fast channel switching based on SVC in IPTV environments". In: *IEEE transactions on broadcasting* 58.1 (2011), pp. 57–65.

[59] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck. "An HTTP/2 push-based approach for low-latency live streaming with super-short segments". In: *Journal of Network and Systems Management* 26.1 (2018), pp. 51–78.

[60] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner. "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming". In: *Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE. 2014, pp. 111–116.

[61] M. Seufert, T. Hoßfeld, and C. Sieber. "Impact of intermediate layer on quality of experience of HTTP adaptive streaming". In: *11th International Conference on Network and Service Management (CNSM)*. IEEE. 2015, pp. 256–260.

[62] T. Hoßfeld, C. Moldovan, and C. Schwartz. "To each according to his needs: Dimensioning video buffer for specific user profiles and behavior". In: *International Symposium on Integrated Network Management (IM)*. IEEE. 2015, pp. 1249–1254.

[63] P. Wisniewski, A. Beben, J. M. Batalla, and P. Krawiec. "On delimiting video rebuffering for stream-switching adaptive applications". In: *International Conference on Communications (ICC)*. IEEE. 2015, pp. 6867–6873.

[64] J. M. Batalla, P. Krawiec, A. Beben, P. Wisniewski, and A. Chydzinski. "Adaptive video streaming: Rate and buffer on the track of minimum rebuffering". In: *IEEE Journal on Selected Areas in Communications* 34.8 (2016), pp. 2154–2167.

[65] P. Juluri, V. Tamarapalli, and D. Medhi. "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP". In: *International Conference on Communication Workshop (ICCW)*. IEEE. 2015, pp. 1765–1770.

[66] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. "A survey on quality of experience of HTTP adaptive streaming". In: *IEEE Communications Surveys & Tutorials* 17.1 (2014), pp. 469–492.

[67] 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: *Fourth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE. 2012, pp. 1–6.

[68] T. Hossfeld, D. Strohmeier, A. Raake, and R. Schatz. "Pippi Longstocking calculus for temporal stimuli pattern on YouTube QoE". In: *Proceedings of the 5th Workshop on Mobile Video*. 2013, pp. 37–42.

[69] S. Egger, B. Gardlo, M. Seufert, and R. Schatz. "The impact of adaptation strategies on perceived quality of http adaptive streaming". In: *Proceedings of the Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. ACM. 2014, pp. 31–36.

[70] C. Moldovan, K. Hagn, C. Sieber, W. Kellerer, and T. Hoßfeld. "Keep calm and don't switch: About the relationship between switches and quality in HAS". In: *29th International Teletraffic Congress (ITC)*. Vol. 3. IEEE. 2017, pp. 1–6.

[71] S. Winkler. "Video quality and beyond". In: *15th European Signal Processing Conference*. IEEE. 2007, pp. 150–153.

[72] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[73] Q. Huynh-Thu and M. Ghanbari. "Scope of validity of PSNR in image/video quality assessment". In: *Electronics letters* 44.13 (2008), pp. 800–801.

[74] J. Y. Lin, T.-J. Liu, E. C.-H. Wu, and C.-C. J. Kuo. "A fusion-based video quality assessment (FVQA) index". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE. 2014, pp. 1–5.

[75] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. Cock. "VMAF: The journey continues". In: *Netflix Technology Blog* 25 (2018).

[76] R. V. Babu, A. S. Bopardikar, A. Perkis, and O. I. Hillestad. "No-reference metrics for video streaming applications". In: *International Workshop on Packet Video*. Vol. 2. sn. 2004, pp. 10–11.

[77] S. Winkler and R. Campos. "Video quality evaluation for Internet streaming applications". In: *Human Vision and Electronic Imaging VIII*. Vol. 5007. International Society for Optics and Photonics. 2003, pp. 104–115.

[78] L. Ma, S. Li, and K. N. Ngan. "Reduced-reference video quality assessment of compressed video sequences". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.10 (2012), pp. 1441–1456.

[79] F. Fotrousi, S. A. Fricker, and M. Fiedler. "The effect of requests for user feedback on Quality of Experience". In: *Software Quality Journal* 26.2 (2018), pp. 385–415.

[80] ITU-T. "ITU-T recommendation P.800.2 : Mean opinion score interpretation and reporting". In: *Technical Report of the International Telecommunication Union*. 2016.

[81] R. C. Streijl, S. Winkler, and D. S. Hands. "Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives". In: *Multimedia Systems* 22.2 (2016), pp. 213–227.

[82] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten. "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1". In: *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[83] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M.-N. Garcia, K. Yamagishi, and S. Broom. "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software". In: *9th ACM Multimedia Systems Conference (MMSys)*. ACM, 2018, pp. 466–471.

[84] M. Seufert, N. Wehner, and P. Casas. "Studying the impact of HAS QoE factors on the standardized QoE model P. 1203". In: *International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 1636–1641.

[85] R. R. Ramachandra Rao, S. Göring, P. Vogel, N. Pachatz, J. J. V. Villarreal, W. Robitza, P. List, B. Feiten, and A. Raake. "Adaptive video streaming with current codecs and formats: Extensions to parametric video quality model ITU-T P. 1203". In: *Electronic Imaging* 10 (2019), pp. 314–1.

[86] H. T. Tran, N. P. Ngoc, T. Hoßfeld, and T. C. Thang. "A cumulative quality model for HTTP adaptive streaming". In: *Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2018, pp. 1–6.

[87] H. T. Tran, N. P. Ngoc, T. Hoßfeld, M. Seufert, and T. C. Thang. "Cumulative Quality Modeling for HTTP Adaptive Streaming". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17.1 (2021), pp. 1–24.

[88] Y.-F. Ou, Y. Xue, and Y. Wang. "Q-STAR: A perceptual video quality model considering impact of spatial, temporal, and amplitude resolutions". In: *IEEE Transactions on Image Processing* 23.6 (2014), pp. 2473–2486.

[89] H. T. Tran, N. P. Ngoc, Y. J. Jung, A. T. Pham, and T. C. Thang. "A histogram-based quality model for HTTP adaptive streaming". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 100.2 (2017), pp. 555–564.

[90] Y. Shen, Y. Liu, Q. Liu, and D. Yang. "A method of QoE evaluation for adaptive streaming based on bitrate distribution". In: *IEEE International Conference on Communications Workshops (ICC)*. IEEE. 2014, pp. 551–556.

[91] H. Ebbinghaus. "Memory: A contribution to experimental psychology". In: *Annals of Neurosciences* 20.4 (2013), p. 155.

[92] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz. ""Time is bandwidth"? Narrowing the gap between subjective time perception and Quality of Experience". In: *International Conference on Communications (ICC)*. IEEE. 2012, pp. 1325–1330.

[93] P. Casas, M. Seufert, S. Egger, and R. Schatz. "Quality of experience in remote virtual desktop services". In: *International Symposium on Integrated Network Management (IM)*. IEEE. 2013, pp. 1352–1357.

[94] S. Zadtootaghaj, S. Schmidt, and S. Möller. "Modeling gaming QoE: Towards the impact of frame rate and bit rate on cloud gaming". In: *Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2018, pp. 1–6.

[95] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. "BOLA: Near-optimal bitrate adaptation for online videos". In: *International Conference on Computer Communications (INFOCOM)*. IEEE. 2016, pp. 1–9.

[96] V. Burger, T. Zinner, L. Dinh-Xuan, F. Wamser, and P. Tran-Gia. "A generic approach to video buffer modeling using discrete-time analysis". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.2s (2018), pp. 1–23.

[97] P. Tran-Gia. *Einführung in die Leistungsbewertung und Verkehrstheorie*. Oldenbourg Wissenschaftsverlag, 2009.

[98] P. Tran-Gia and T. Hoßfeld. *Performance Modeling and Analysis of Communication Networks: A Lecture Note*. BoD–Books on Demand, 2021.

[99] L. Kleinrock. *Queueing systems: Theory*. John Wiley, 1975.

[100] L. De Cicco, G. Cofano, and S. Mascolo. "A hybrid model of the akamai adaptive streaming control system". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 3321–3326.

[101] A. Anttonen and A. Mammelä. "Interruption probability of wireless video streaming with limited video lengths". In: *IEEE Transactions on Multimedia* 16.4 (2014), pp. 1176–1180.

[102] T. H. Luan, L. X. Cai, and X. Shen. "Impact of network dynamics on user's video quality: Analytical framework and QoS provision". In: *IEEE Transactions on Multimedia* 12.1 (2009), pp. 64–78.

[103] A. ParandehGheibi, M. Médard, A. Ozdaglar, and S. Shakkottai. "Avoiding interruptions – A QoE reliability function for streaming media applications". In: *IEEE Journal on Selected Areas in Communications* 29.5 (2011), pp. 1064–1074.

[104] Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, and T. Jimenez. "Probabilistic analysis of buffer starvation in Markovian queues". In: *International Conference on Computer Communications (INFOCOM)*. IEEE. 2012, pp. 1826–1834.

[105] F. Yu, H. Chen, L. Xie, and J. Li. "Impact of end-user playout buffer dynamics on http progressive video qoe in wireless networks". In: *25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. IEEE. 2014, pp. 1996–2001.

[106] P. K. Yadav, A. Shafiei, and W. T. Ooi. "QUETRA: A queuing theory approach to dash rate adaptation". In: *Proceedings of the 25th ACM International Conference on Multimedia*. ACM. 2017, pp. 1130–1138.

[107] K. Gatimu and B. Lee. "qMDP: DASH Adaptation using Queueing Theory within a Markov Decision Process". In: *18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2021, pp. 1–6.

[108] A. Bentaleb, P. K. Yadav, W. T. Ooi, and R. Zimmermann. "DQ-DASH: A Queuing Theory Approach to Distributed Adaptive Video Streaming". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16.1 (2020), pp. 1–24.

[109] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia. "Performance modeling of softwarized network functions using discrete-time analysis". In: *28th International Teletraffic Congress (ITC)*. Vol. 1. IEEE. 2016, pp. 234–242.

[110] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms". In: *Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. 2014, pp. 1–6.

[111] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. "Commute path bandwidth traces from 3G networks: analysis and applications". In: *Proceedings of the Multimedia Systems Conference (MMSys)*. ACM. 2013, pp. 114–118.

[112] A. M. Law, W. D. Kelton, and W. D. Kelton. *Simulation modeling and analysis*. Vol. 3. McGraw-Hill New York, 2000.

[113] M. Manohara, A. Moorthy, J. De Cock, I. Katsavounidis, and A. Aaron. *Optimized shot-based encodes: Now Streaming!* Netflix Inc. 2018. URL: `https://medium.com/netflix-techblog/optimized-shot-based-encodes-now-streaming-4b9464204830` (visited on Sept. 20, 2021).

[114] O. Zach and M. Slanina. "Content aware segment length optimization for adaptive streaming over HTTP". In: *Radio Engineering* 27.3 (2018), p. 819.

[115] V. Adzic, H. Kalva, and B. Furht. "Optimizing video encoding for adaptive streaming over HTTP". In: *IEEE Transactions on Consumer Electronics* 58.2 (2012), pp. 397–403.

[116] A. Zabrovskiy, C. Feldmann, and C. Timmerer. "A practical evaluation of video codecs for large-scale HTTP adaptive streaming services". In: *International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 998–1002.

[117] Z. Wang and A. C. Bovik. "A universal image quality index". In: *IEEE Signal Processing Letters* 9.3 (2002), pp. 81–84.

[118] X. Min, G. Zhai, J. Zhou, M. C. Farias, and A. C. Bovik. "Study of subjective and objective quality assessment of audio-visual signals". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 6054–6068.

[119] A. Sideris, E. Markakis, N. Zotos, E. Pallis, and C. Skianis. "MPEG-DASH users' QoE: The segment duration effect". In: *Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. Costa Navarino, Messinia, Greece: IEEE, 2015, pp. 1–6.

[120] S. Lederer, C. Müller, and C. Timmerer. "Dynamic adaptive streaming over HTTP dataset". In: *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*. Chapel Hill, North Carolina, USA: ACM, 2012, pp. 89–94.

[121] Y.-T. Lin, T. Bonald, and S. E. Elayoubi. "Impact of chunk duration on adaptive streaming performance in mobile networks". In: *Wireless Communications and Networking Conference*. Doha, Qatar: IEEE, 2016, pp. 1–6.

[122] C. Liu, I. Bouazizi, and M. Gabbouj. "Segment duration for rate adaptation of adaptive HTTP streaming". In: *International Conference on Multimedia and Expo*. Barcelona, Spain: IEEE, 2011, pp. 1–4.

[123] J. van der Hooft, D. Pauwels, C. De Boom, S. Petrangeli, T. Wauters, and F. De Turck. "Low-latency delivery of news-based video content". In: *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys)*. Amsterdam, The Netherlands: ACM, 2018, pp. 537–540.

[124] I. Irondi, Q. Wang, and C. Grecos. "Optimized adaptation algorithm for HEVC/H.265 dynamic adaptive streaming over HTTP using variable segment duration". In: *Real-Time Image and Video Processing*. Vol. 9897. Brussels, Belgium: International Society for Optics and Photonics, 2016, pp. 185–194.

[125] A. Reed and B. Klimkowski. "Leaky streams: Identifying variable bitrate DASH videos streamed over encrypted 802.11n connections". In: *Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2016, pp. 1107–1112.

[126] S. Xu, S. Sen, Z. M. Mao, and Y. Jia. "Dissecting VOD services for cellular: Performance, root causes and best practices". In: *Proceedings of the Internet Measurement Conference (IMC)*. ACM. 2017, pp. 220–234.

[127] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue. "ABR streaming of VBR-encoded videos: characterization, challenges, and solutions". In: *Proceedings of the International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. ACM. 2018, pp. 366–378.

[128] J. De Cock, Z. Li, M. Manohara, and A. Aaron. "Complexity-based consistent-quality encoding in the cloud". In: *International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1484–1488.

[129] B. Ghojogh and M. Crowley. "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial". In: *arXiv preprint arXiv:1905.12787* (2019).

[130] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[131] C. M. Bishop. *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[132] H. Drucker, C. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik. "Support Vector Regression Machines". In: *Advances in Neural Information Processing Systems*. 1997, pp. 155–161.

[133] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[134] M. Naser and A. Alavi. "Insights into performance fitness and error metrics for machine learning". In: *arXiv preprint arXiv:2006.00887* (2020).

[135] J. Xie, F. R. Yu, T. Huang, et al. "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges". In: *IEEE Communications Surveys & Tutorials* 21 (Aug. 2018), pp. 393–430.

[136] R. Boutaba, M. A. Salahuddin, N. Limam, et al. "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities". In: *Journal of Internet Services and Applications* 9 (Dec. 2018), pp. 16–115.

[137] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li. "Stream-based Machine Learning for Real-time QoE Analysis of Encrypted Video Streaming Traffic". In: *Conference on Innovation in Clouds, Internet and Networks (ICIN)*. IEEE. 2019.

[138] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki. "Measuring video QoE from encrypted traffic". In: *Internet Measurement Conference (IMC)*. ACM. 2016, pp. 513–526.

[139] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li. "I See What you See: Real Time Prediction of Video Quality from Encrypted Streaming Traffic". In: *Internet-QoE*. ACM. 2019, pp. 1–6.

[140] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura. "eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic". In: *Network Traffic Measurement and Analysis Conference (TMA)*. IEEE. 2018.

[141] M. Khokhar, T. Ehlinger, and C. Barakat. "From Network Traffic Measurements to QoE for Internet Video". In: *IFIP Networking Conference*. 2019.

[142] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. "A machine learning approach to classifying YouTube QoE based on encrypted network traffic". In: *Multimedia tools and applications* (2017).

[143] I. Oršolić, P. Rebernjak, M. Sužnjević, and L. Skorin-Kapov. "In-network QoE and KPI monitoring of mobile YouTube traffic: Insights for encrypted iOS flows". In: *14th International Conference on Network and Service Management (CNSM)*. IEEE. 2018, pp. 233–239.

[144] Y. Lin, E. Oliveira, S. Jemaa, and S. Elayoubi. "Machine learning for predicting QoE of video streaming in mobile networks". In: *International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–6.

[145] T. Begluk, J. B. Husić, and S. Baraković. "Machine learning-based QoE prediction for video streaming over lte network". In: *17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. IEEE. 2018, pp. 1–5.

[146] T. Mangla, E. Halepovic, E. Zegura, and M. Ammar. "Drop the packets: using coarse-grained data to detect video performance issues". In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. 2020, pp. 71–77.

[147] M. E. Morocho-Cayamcela, H. Lee, and W. Lim. "Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions". In: *IEEE Access* 7 (2019), pp. 137184–137206.

[148] Y. Wang, P. Li, L. Jiao, Z. Su, N. Cheng, X. S. Shen, and P. Zhang. "A data-driven architecture for personalized QoE management in 5G wireless networks". In: *IEEE Wireless Communications* 24 (2016), pp. 102–110.

[149] A. Martin, J. Egaña, J. Flórez, J. Montalbán, I. G. Olaizola, M. Quartulli, R. Viola, and M. Zorrilla. "Network resource allocation system for QoE-aware delivery of media services in 5G networks". In: *IEEE Transactions on Broadcasting* 64.2 (2018), pp. 561–574.

[150] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin. "Improving traffic forecasting for 5G core network scalability: A Machine Learning approach". In: *IEEE Network* 32 (Nov. 2018), pp. 42–49.

[151] M. Polese, R. Jana, V. Kounev, K. Zhang, S. Deb, and M. Zorzi. "Machine learning at the edge: A data-driven architecture with applications to 5G cellular networks". In: *Transactions on Mobile Computing* (2020).

[152] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang. "Intelligent resource scheduling for 5G radio access network slicing". In: *IEEE Transactions on Vehicular Technology* (2019).

[153] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang. "Deep reinforcement learning for resource management in network slicing". In: *IEEE Access* 6 (2018), pp. 74429–74441.

[154] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez. "DeepCog: Cognitive network management in sliced 5G networks with deep learning". In: *Conference on Computer Communications (INFOCOM)*. IEEE. 2019, pp. 280–288.

[155] C. Hernández-Chulde and C. Cervelló-Pastor. "Intelligent optimization and machine learning for 5G network control and management". In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2019, pp. 339–342.

[156] A. Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[157] S. Kosta, A. Mei, and J. Stefa. "Small world in motion (SWIM): Modeling communities in ad-hoc mobile networking". In: *7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE. 2010, pp. 1–9.

[158] I. Rec. "ITU-T G.107 - The E-model: a computational model for use in transmission planning". In: *International Telecommunication Union* 8.20 (2015).

[159] T. Michael, G. Mittag, and S. Möller. "Analyzing the Fullband E-model and Extending it for Predicting Bursty Packet Loss". In: *Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2020, pp. 1–6.

[160] M. Soloducha and A. Raake. "Speech quality of VoIP: bursty packet loss revisited". In: *Speech Communication; 11. ITG Symposium*. VDE. 2014, pp. 1–4.

[161] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[162] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*. Vol. 11700. Springer Nature, 2019.

[163] A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Müller, and P. Kessel. "Explanations can be manipulated and geometry is to blame". In: *Advances in Neural Information Processing Systems*. 2019.