UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Md Tariqul Islam

A Predictive DASH QoE Approach Based on Machine Learning at Multi-access Edge Computing

Uma abordagem preditiva de DASH QoE baseada em Aprendizado de Máquina em Multi-access Edge Computing

Campinas
2021

Md Tariqul Islam

A Predictive DASH QoE Approach Based on Machine Learning at Multi-access Edge Computing

Uma abordagem preditiva de DASH QoE baseada em Aprendizado de Máquina em Multi-access Edge Computing

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de Computação.

Supervisor/Orientador: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Este trabalho corresponde à versão final da dissertação defendida pelo aluno Md Tariqul Islam, orientada pelo Prof. Dr. Christian Rodolfo Esteve Rothenberg.

Campinas
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Islam, Md Tariqul, 1993-
Is4p A predictive DASH QoE approach based on machine learning at multi-access edge computing / Md Tariqul Islam. – Campinas, SP : [s.n.], 2021.

Orientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Vídeos para internet. 2. Experiencia. 3. Aprendizado de máquina. 4. Redes de computadores. 5. Arquitetura de redes de computador. I. Rothenberg, Christian Rodolfo Esteve, 1982-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Uma abordagem preditiva de DASH QoE baseada em aprendizado de máquina em multi-access edge computing
Palavras-chave em inglês:
Internet videos
Experience
Machine learning
Computer network
Network architecture
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Christian Rodolfo Esteve Rothenberg [Orientador]
Michele Nogueira de Lima
Roberto Riggio
Data de defesa: 05-03-2021
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-9130-8606
- Currículo Lattes do autor: http://lattes.cnpq.br/2920555284231309

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Md Tariqul Islam RA: 228009
Data da Defesa: 5 de março de 2021
Título da Tese: "A Predictive DASH QoE Approach Based on Machine Learning at Multi-access Edge Computing"
Título em outro idioma: "Uma abordagem preditiva de DASH QoE baseada em Aprendizado de Máquina em Multi-access Edge Computing"

Prof. Dr. Christian Rodolfo Esteve Rothenberg (FEEC/UNICAMP) - Presidente
Profa. Dra. Michele Nogueira de Lima (UFPR/Universidade Federal do Paraná)
Prof. Dr. Roberto Riggio (RISE/Research Institutes of Sweden)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

I dedicate this master's thesis to my family, relatives, and friends for their constant inspiration and support.

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisor Prof. Dr. Christian Rothenberg, who trusted my abilities and guided me throughout my master's journey to prepare this thesis.

I am grateful to Dr. Jason Quinlan from Ireland, who helped me a lot by replying to my emails to ease my experimental setup.

I would like to thank each of my family members and close relatives, who always supported and inspired me, despite being far away, to continue my studies and make this thesis possible.

I am thankful to all friends and colleagues from the INTRIG for always being by my side. Specifically, I have to mention Rony, Raza, and Meer's names with whom I spent most of my time. Moreover, Danny and Nathan's name, whose sincere cooperation helped me adapt to the control plane project.

Finally, I am grateful to Innovation Center, Ericsson, Brazil, for the financial and technical support.

"Done is better than perfect."
-Sheryl Sandberg

Abstract

Multimedia video services traffic is rapidly growing in mobile networks in recent years. Video services using Dynamic Adaptive Streaming over HTTP (DASH) techniques have dominated the total internet traffic to carry video traffic. Mobile Network Operators (MNOs) are expected to run on with this growing demand for DASH-supported video traffic while providing a high Quality of Experience (QoE) to the end-users. Besides, operators need to have a crystal notion of video quality perceived by the end-users and correlate them with network-level monitoring or telemetry information for problem identification, root cause analysis, and pattern prediction. To ensure QoE–aware network traffic management, a prerequisite for the MNOs is to monitor the network traffic passively and measure objective QoE Key Performance Indicators (KPIs) (such as resolutions and stalling events) effectively that directly influence end-user subjective feedback. Many literature approaches have been proposed to measure the KPIs aimed to deliver acceptable video service quality. Most of the solutions require end-user awareness, which is not viable from the MNOs' perspective. However, Deep Packet Inspection (DPI), another most widely used solution to estimate the KPIs directly from network traffic, is not a convenient solution anymore for the operators due to the adoption of end-to-end video streaming encryption over TCP (HTTPs) and QUIC transport protocol. Hence, in recent, Machine Learning (ML) has been accepted as a well-recognized solution for estimating QoE KPIs by analyzing the encrypted traffic patterns and statistics as Quality of Service (QoS).

This work presents an ML-based lightweight and fine-grained `Edge QoE Probe` approach to estimate the end-user QoE for DASH video service by passively monitoring the encrypted network traffic on the edge of the network. Our approach can assess numerous QoE KPIs (such as resolution, bit-rate, quality switches, startup delay, and stall ratio) both in a real-time and per-session manner. Moreover, we investigate the DASH video service performance over the traditional TCP (HTTPs) and QUIC transport protocol in this work. For this purpose, we experimentally evaluate different cellular network traces in a high-fidelity emulated testbed and compare the behavioral performance of Adaptive Bitrate Streaming (ABS) algorithms considering QoE KPIs over TCP (HTTPs) and QUIC. Our empirical results show that QUIC suffers from traditional state-of-the-art ABS algorithms' ineffectiveness to improve video streaming performance without specific changes.

Keywords: Dynamic Adaptive Streaming over HTTP (DASH); Quality of Experience (QoE); Quality of Service (QoS); Machine Learning (ML); Edge Computing (EC); Transport Protocol.

Resumo

O tráfego de serviços de vídeo multimídia está crescendo rapidamente nas redes móveis nos últimos anos. Os serviços de vídeo que usam técnicas de Dynamic Adaptive Streaming sobre HTTP (DASH) dominaram o tráfego total da Internet para transportar o tráfego de vídeo. Espera-se que as operadoras de rede móvel (Mobile Network Operators - MNOs) continuem atendendo a essa demanda crescente por tráfego de vídeo suportado por DASH, ao mesmo tempo em que fornecem uma alta qualidade de experiência (Quality of Experience - QoE) aos usuários finais. Além disso, as operadoras precisam ter um conhecimento claro acerca da qualidade de vídeo percebida pelos usuários finais e relacioná-la com o monitoramento em nível de rede, ou com informações de telemetria para identificação de problemas, análise da causa raiz e predição de padrões. Para garantir um gerenciamento de tráfego de rede com reconhecimento de QoE, um pré-requisito é que os MNOs monitorem o tráfego de rede passivamente e realizem medições efetivas de indicadores-chave de desempenho (Key Performance Indicators - KPIs) de QoE, como resoluções, eventos de paralisação, entre outros, que influenciam diretamente a percepção do usuário final. Muitas abordagens da literatura foram propostas para medir os KPIs com o objetivo de fornecer uma qualidade de serviço de vídeo aceitável. A maioria das soluções exige consciência de contexto do usuário final, o que não é viável do ponto de vista do MNO. No entanto, Deep Packet Inspection (DPI), outra solução mais amplamente usada para estimar os KPIs diretamente do tráfego de rede, não é mais uma solução conveniente para as operadoras devido à adoção de criptografia de streaming de vídeo fim-a-fim sobre TCP (HTTPs) e QUIC. Portanto, o aprendizado de máquina (Machine Learning - ML) passou a ser recentemente aceito como uma solução bem reconhecida para estimar KPIs de QoE, analisando os padrões de tráfego criptografados bem como estatísticas como qualidade de serviço (Quality of Service - QoS).

Este trabalho apresenta uma abordagem mais refinada e leve, baseada em aprendizado de máquina, denominada `Edge QoE Probe`, para estimar QoE do usuário final para o serviço de vídeo DASH, monitorando passivamente o tráfego de rede criptografado na borda da rede. Nossa abordagem pode avaliar vários KPIs de QoE, como por exemplo resolução, taxa de bits, proporção de paralisação, entre outros, tanto em tempo real quanto por sessão. Além disso, neste trabalho investigamos o desempenho do vídeo DASH sobre o protocolo de transporte tradicional TCP (HTTPs) e QUIC. Para este propósito, avaliamos experimentalmente diferentes traces de rede celular em um ambiente emulado de alta fidelidade e comparamos o desempenho comportamental de algoritmos Adaptive Bitrate Streaming (ABS) considerando KPIs de QoE sobre TCP (HTTPs) e QUIC. Nossos resultados empíricos mostram que os algoritmos tradicionais de ABS usando QUIC como transporte precisariam alterações específicas para melhorar o desempenho em termos de QoE de vídeo baseados em DASH.

Palavras-chave: Dynamic Adaptive Streaming over HTTP (DASH); Qualidade de Experiência (QoE); Qualidade de Serviço (QoS); Aprendizado de Máquina (ML); Computação de Borda (EC); Protocolo de Transporte.

List of Figures

List of Tables

Acronyms

ABS Adaptive Bitrate Streaming.

CCL Closed Control Loop.

CDNs Content Delivery Networks.

DASH Dynamic Adaptive Streaming over HTTP.

DPI Deep Packet Inspection.

EC Edge Computing.

ETSI European Telecommunications Standards Institute.

HAS HTTP Adaptive Streaming.

HOL Head of Line.

HTTPs HyperText Transfer Protocol Secure.

IETF Internet Engineering Task Force.

MANO Management and Orchestration.

MEC Multi-access Edge Computing.

ML Machine Learning.

MNOs Mobile Network Operators.

MOS Mean Opinion Score.

MPD Media Presentation Description.

NFV Network Functions Virtualization.

NFVI Network Function Virtualization Infrastructure.

OTT Over-the-Top.

OvS Open vSwitch.

QoE Quality of Experience.

QoS Quality of Service.

QUIC Quick UDP Internet Connections.

RAN Radio Access Network.

SDN Software-Defined Networking.

SLA Service-Level Agreement.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

VNF Virtual Network Function.

VoD Video-on-Demand.

Contents

Chapter 1

Introduction

Over the past year, vast development in telecommunication and networking areas has fostered global internet traffic growth. With the rapid evolution of electronic gadgets and multimedia services, the number of internet users worldwide is forecasted to reach 5.3 billion by 2023, as shown in Figure 1.1, up from 3.9 billion in 2018 [1]. Hence, in terms of the global population, two-thirds will have internet access by the end of 2023—such an increased rate of internet users creates massive traffic while using diverse applications over the internet.

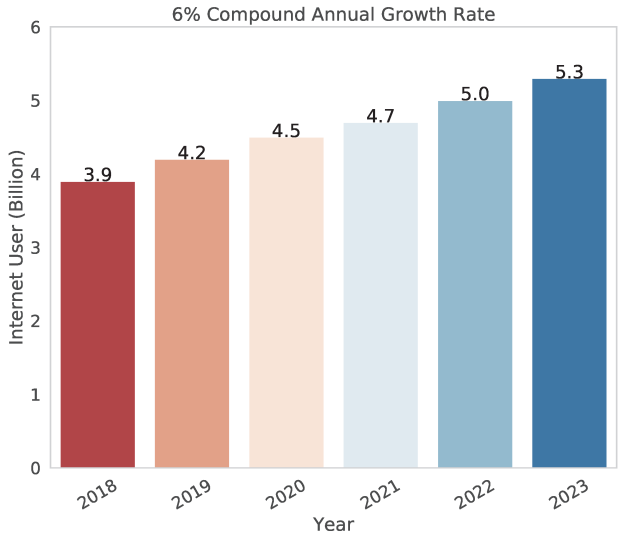


Figure 1.1: Global Internet User Growth

According to the Global Internet Phenomena Report [2], the downstream video traffic accounted for 65% of global internet traffic in 2020, mainly derived from popular video streaming services (e.g., YouTube and Netflix) and social networks (e.g., Facebook and Instagram) in the multimedia application domain. In the Cisco VNI whitepaper [3], video streaming traffic accounted for 60% of the total internet traffic in 2017. By 2022, it has expected to reach 82% of the total internet traffic with immense video traffic growth rates. Besides, Figure 1.2 depicts the distribution of global internet traffic for diverse applications from 2016 to 2021, based on the earlier Cisco VNI whitepaper [4]. It indicates that internet[1] video traffic has always been

[1]Includes short-form Internet video (for example, YouTube), long-form Internet video (for example, Hulu), live Internet video, Internet video to TV (for example, Netflix through Roku), online video purchases and rentals, webcam viewing, and web-based video monitoring (excludes P2P video file downloads)

used at a significant rate. On the other hand, according to Ericson Mobility Report [5], video content traffic accounted for 63% of global mobile traffic in 2019 and is projected to reach 76% by 2025. The recent pandemic due to COVID-19 increased the video content watching time by 60% in early 2020 [6]. The essence of all the reports shows the predominance of internet video application traffic in the foreseeable future.

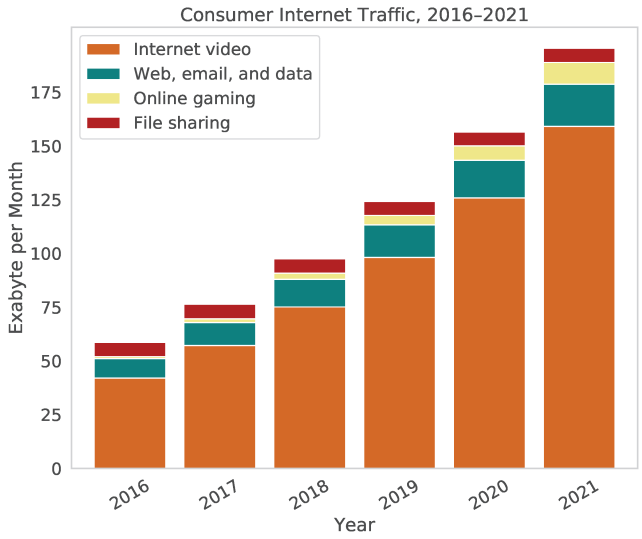


Figure 1.2: The Distribution of Global Internet Traffic for Diverse Applications

Moreover, these days, a large number of internet video streaming is provided via Over-the-Top (OTT) platforms. Hence, the popularity of OTT platforms such as Netflix, Hulu, Amazon, and YouTube/YouTube Red is growing enormously for Video-on-Demand (VoD) services. Where HTTP Adaptive Streaming (HAS) is considered a de-facto standard to carry video traffic for VoD services [7]. Therefore, it has become a critical issue for stakeholders such as Content Providers, Content Delivery Networks (CDNs), and Mobile Network Operators (MNOs) to manage such video traffic demand and ensure that end-users perceive the best quality. They all want to provide a satisfactory experience to their end-users, known as Quality of Experience (QoE). To ensure better QoE, for each stakeholder, specifically for MNOs, understanding and monitoring the key factors that impact users' perceived experience and service quality has become challenging. It is also a trending topic in QoE-related research among the networking community in recent years [8]. The goal of such research is to introduce QoE-driven strategies into the network management system [9].

The limitation in traditional networks (e.g., flexibility, agility, and scalability) for ongoing QoE monitoring and management, fostering future networks (e.g., 5G and Beyond) towards softwarization and virtualization via Software-Defined Networking (SDN) [10] and Network Functions Virtualization (NFV) [11]. The NFV role will be decoupling the network functions as a virtual function from the hardware appliance. The SDN will enable the NFV networking infrastructure's programmability to treat the control and forwarding planes separately. Besides, the network resources will be equipped with cutting-edge technologies such as Edge Computing (EC) to decentralize the processing power for customizing the delivery service and meet the end-user need for the intended QoE via intelligent QoE monitoring and management techniques. Therefore, QoE aware/driven strategies for HAS-based video service in SDN and NFV context, specifically EC as emerging architecture, are current research questions.

1.1 Motivation

Existing QoE monitoring system taking end-user participation. Typically to assess the QoE for any specific multimedia service (e.g., video streaming), active users contribute with subjective feedback to evaluate the given service, which is quite expensive in terms of time, money, and manual efforts [8]. Due to this fact, the alternative of subjective assessment is to calculate the objective QoE from the information collected at the application level, allowing the installation of specific tools (e.g., software or application) on the user's terminal [12] [13]. However, this approach has limited wide-applicability and required end-user engagement to estimate user-level QoE as well as annoying to the user always asking about the service (e.g., video streaming). Most importantly, such tools provided user end QoE information is not accessible by the MNOs for taking adequate action to overcome the quality degradation.

Traditional passive Deep Packet Inspection (DPI). Usually, content providers and CDNs directly measure end-user QoE either by using logs from server-side or client-side tools. In contrast, to assess the end-user perceived quality, network operators (e.g., MNOs) only rely on the traffic as it passes through the network. Traditionally, network operators rely on on-path middle-box processing. Which is a passive DPI monitoring probe technique to examine the HTTP packet flows. It extracts the video session information to estimate the application-level QoE metrics (e.g., startup delay, re-buffering events, resolutions) [14] [15]. However, according to European Union (EU) General Data Protection Regulation (GDPR) compliance, such a DPI-based solution has limited applicability. Due to data transfer privacy concerns, in recent OTT platforms delivering their video streaming services with encryption. At this moment, contents are started serving over HyperText Transfer Protocol Secure (HTTPs), a combination of HTTP+TLS+TCP. An alternative of HTTPs due to TCP transport's native shortcomings for earlier HTTP versions, recently Google introduced QUIC transport that incorporates TLS by default. Moreover, QUIC has been adopted as a new transport protocol for the newer HTTP (HTTP/3) version at a recent time [16]. End-to-end encryption due to HTTPs and QUIC makes MNO blackout look at the video session quality directly from network traffic. Network operators no longer have access to extract the application-level QoE metrics from HTTP packet flow, as shown in Figure 1.3.

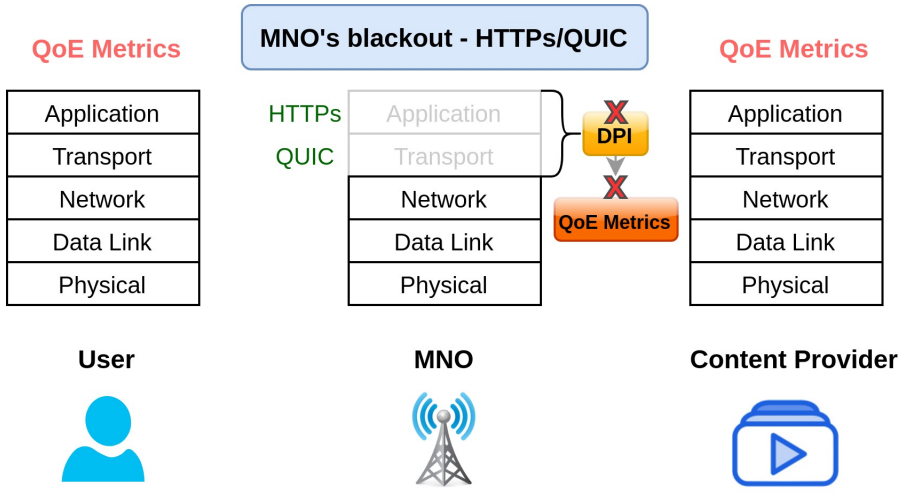


Figure 1.3: End-to-End Encryption over HTTPs and QUIC (adopted from [17])

Thus, in recent the network operators rely on IP packet patterns and their statistical characteristics to predict the application-level QoE metrics and the potential root cause that leads

to degrading quality. Several works recently leveraged Machine Learning (ML) techniques to map network traffic patterns with application-level QoE metrics. They found that network-level Quality of Service (QoS) metrics (e.g., bandwidth, delay, packet loss) directly impact QoE. However, most of the mapping methods are not lightweight, fine-grained, and generalized.

Adaptive streaming performance over TCP and QUIC transport protocol. HTTP Adaptive Streaming (HAS) is considered the de-facto standard to deliver VoD services, and Dynamic Adaptive Streaming over HTTP (DASH) is regarded as the most dominating format for implementing HAS [18]. In this form, the Adaptive Bitrate Streaming (ABS) algorithm is deployed in a client-side player that detects a favorable quality of the video stream between multiple bit-rates and resolutions by adapting the network changes. Therefore, end-user receives interrupt-free video streaming services that enhance overall QoE. The ABS algorithms were initially designed and built over TCP transport as the HTTP standard requires a reliable transport [19]. As a result, HAS has been using the TCP transport protocol predominantly for many years. Despite the reliability and in-order delivery benefits of TCP, both HTTP/1.1 and HTTP/2 versions of application-level HTTP standard over traditional TCP transport suffers from Head of Line (HOL) blocking problem [20]. Moreover, for a secure connection, TLS over TCP requires an additional handshake latency. In contrast, for the third version of the HTTP standard (HTTP/3) [16], the new transport protocol QUIC [21] running over the UDP solves the issues of HOL blocking. Besides, newly standardized QUIC transport embraces other features such as fast connection establishment, improving congestion control, forward error correction, and seamless connection migration. The aforementioned motivates us to evaluate the impact of QUIC transport on QoE for HAS (e.g., DASH) video service. Specifically, to assess state-of-the-art ABS algorithms (built on TCP concept) performance over QUIC transport.

Inadequate QoE monitoring and optimization work in the scope of Edge Computing (EC). HAS (e.g., DASH) standard video service prone to adjust video quality based on network function status. Therefore, the MNO must be aware of network conditions and the capacity to ensure acceptable video quality by optimizing network performance on run time. However, to meet the vast quantity of video service demand and provide the most desirable QoE in the foreseeable future has to lead advent cutting-edge technologies such as EC. It brings real-time computing processing for low latency, high throughput, and reliability features, as well as data and applications close to the end-user. To implement the EC concept, European Telecommunications Standards Institute (ETSI) defined the Multi-access Edge Computing (MEC) [22] paradigm providing an IT service environment and cloud-computing capabilities at the edge of the mobile network, integrating the Radio Access Network (RAN). A couple of work has been done in recent times with the MEC paradigm in multimedia service. Most of the work presents the benefits of adding a MEC server to raise the end-user QoE in the context of content caching, bit-rate optimization, and service migration. Nevertheless, no significant works have not been done in the scope of MEC potential use cases [23], under *"Network Performance and QoE Improvement Services,"* to optimize the network performance by monitoring network traffic. In such a use case, a QoE aware probe solution, as shown in Figure 1.4, will be responsible for monitoring the fronthaul (e.g., RAN) or backhaul (e.g., core, internet) network traffic and estimating the end-user QoE. MEC also extends the notion of the NFV for virtualization and SDN for QoE management approaches; hence, such a QoE probe would act as a Virtual Network Function (VNF) probe (vProbe). Following the estimated QoE information would help the MEC orchestrator (e.g., cloud edge SDN controller) in taking appropriate action (e.g., optimize the backhaul transport) to boost the network performance.

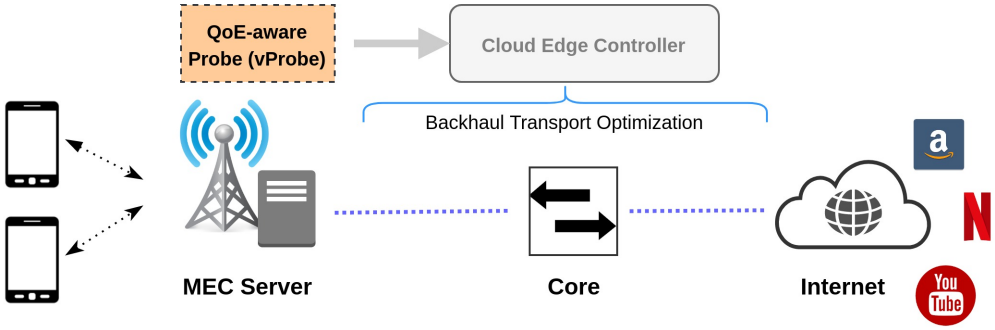


Figure 1.4: A High-level Overview of QoE-aware MEC Solution

1.2 Aim and Scope

This work aims to propose a lightweight approach for **a predictive QoE probe scheme to allow the network operators to infer users' QoE at the network edge facility for adaptive video streaming service (e.g., DASH)**. The edge node will act as sinks of target end-users network monitoring traffic flows. The analysis of the network traffic flows collected and processed by the QoE probe scheme at the edge will allow the network operator to estimate end-user objective QoE metrics (e.g., resolution, bit-rate, and stall). The reasons to select the edge node for the sink of target end-users network traffic are- *First*, nearest the monitoring location provides the closest prediction of end-user perceived video service quality. *Second*, traffic monitoring at other locations (e.g., core) in network premises, away from the user, can lead to error-prone predictions due to cross-traffic interference.

The outcome for such end-user QoE prediction from network-level measurement has a two-fold impact on QoE-driven network management:

1. **Proactive: Per-session QoE estimation and Service-Level Agreement (SLA).** SLA defines the level of service a user expects from a network operator; therefore, the per video session QoE estimation store in SLA helps network operators evaluate their delivery service for proactive QoE optimization and better network capacity planning.

2. **Reactive: Real-time QoE estimation and autonomous Closed Control Loop (CCL).** A service assurance CCL system continuously assesses real-time, i.e., a specific time window's network conditions, traffic demands, and resource availability. In autonomous CCL, network orchestrator (e.g., SDN controller) plays a role in taking the appropriate action based on the assessment. Hence, real-time QoE estimation helps online service assurance CCL periodically (e.g., on a specific time window basis) correlate network information with estimated objective QoE metrics to determine the root cause and predict QoE impairment patterns. Thus, the orchestrator can take run time action (e.g., reroute or reshape traffic) to improve user QoE.

Specific requirements that guide this work include:

- **Design and implementation of predictive QoE probe scheme to deal with a softwarized intelligent-driven network.** The predictive QoE probe scheme should be feasible in traffic monitoring and QoE inferring to deal with the EC concept. Therefore, in the context of virtualization and softwarization, the QoE probe scheme can fit as a virtual function (vProbe), and QoE analytics can deal with a softwarized (e.g., SDN-based architecture) intelligent-driven network for run-time network performance optimization.

- **Understanding the video streaming performance over different transport protocols.** Imposing end-to-end encryption for video streaming service (e.g., DASH) over TCP (HTTPs) and QUIC transport requires knowing the notion of video streaming performance for different video streaming schemes (e.g., ABS algorithms). Moreover, the newly standardized QUIC transport protocol promoted to obtain numerous advantages over TCP. Therefore, the QoE probe scheme should be aware of the state-of-the-art ABS algorithm's behavior over traditional TCP (HTTPs) and QUIC transport.

In order to achieve the main goal of this thesis, the following specific objectives have been defined:

- **No end-user cooperation.** To estimate QoE, network operators (e.g., MNOs) will not need to rely on users directly to get feedback or installing any tools in the end-user terminal. Our proposed approach will allow the network operator to estimates user-end application-level QoE by passive network-level traffic monitoring at the edge of the network without annoying the end-user.

- **Lightweight, fine-grained method without requiring DPI and segment information.** The QoE probe scheme will not need on-path complex computational processing such as DPI. QoE prediction/estimation will be based on lightweight network features (e.g., QoS) that rely on encrypted IP packet header information without computationally expensive video segment identification. The prediction method (e.g., ML-based QoS-to-QoE correlation model) will be fine-grained (both for real-time and per-session) instead of coarse granularity (e.g., per-session good or bad QoE).

- **Experimental evaluation of QoE estimation and performance under realistic scenarios.** This work will consider realistic scenarios to conduct a large-scale experimental assessment, with an emulation-based DASH-supported player, including end-to-end encryption via HTTPs and QUIC and diverse mobility patterns derived from different cellular network technologies throughput/bandwidth information. Such realistic throughput/bandwidth information will be fruitful for evaluating the QoE performance of different ABS algorithms.

1.3 Contributions

1. We design a non-invasive network-level encrypted traffic measurement method based on passive probing running on edge computing facility (e.g., MEC) to estimate video QoE metrics without requiring endpoints awareness and on-path middlebox processing. Such QoE estimation we define as `Edge QoE Probe` tailored to specific network region (e.g., edge), network topology, network condition (e.g., congestion/bottleneck in specific backhaul links), end-user (e.g., individual user or group), and service (e.g., DASH video).

2. We propose a lightweight, fine-grained network-level temporal QoS features extraction technique stand on the three-time window, i.e., *current*, *trend*, and *session*, by observing bi-directional encrypted network packets' IP header information. We set the shortest granularity time window of 0.5-second length to compute different statistics of temporal QoS features. By analyzing the extracted temporal QoS features, we estimate real-time (e.g., 0.5-second time interval) fine-grained (e.g., multi-class classification) quality of displayed DASH video QoE metrics such as resolution and bit-rate instead of coarse-grained

estimation (e.g., binary classification) to enable quick reactive QoE management service. We further show the per-session level QoE metrics estimation approach by aggregating the QoS features computed at the *current* time window. We estimate per-session fine-grained (e.g., multi-class classification) QoE metrics such as average resolution, average bit-rate, quality switches, stall ratio, and average MOS based on ITU-T P.1203. Apart from classification techniques, we provide a more fine-grained continuous estimation of the average MOS and startup delay using regression techniques. Such per-session QoE estimation enables proactive QoE management service.

3. To estimate the QoE metrics by analyzing QoS features through the QoS-to-QoE correlation model, we present different ML algorithms' benchmark. We evaluate the different models' performance based on their prediction accuracy and training time to pick the most suitable algorithm for our QoS-to-QoE correlation model. Moreover, we study more on QoS features by their relative importance on specific QoE metric prediction.

4. We show that `Edge QoE Probe` performs nearly accurately to estimate video QoE metrics for real-time and per-session over our generated datasets. To the best of our knowledge, we are the first to use a high-fidelity and fully controllable Mininet-WiFi-based emulation environment for the sake of QoE estimation. We use a heterogeneous combination to generate the dataset, i.e., three cellular network technologies (3G, 4G, and 5G) traces from different mobility to set link conditions, two transport protocols (QUIC and TCP), six state-of-the-art ABS algorithms using goDASH player. goDASH is a lightweight headless DASH video player built mainly for emulation environment to conduct large-scale adaptive video streaming evaluation.

5. We evaluate the DASH video service's QoE performance over traditional TCP and open-source implementation of newly standardized QUIC transport protocol. With our empirical QoE study, we try to answer whether QUIC has been able to keep its promise of better QoE for unmodified state-of-the-art ABS algorithm on diverse (e.g., stable and unstable) network conditions.

6. We present a reproducible artifact to conduct large-scale emulation-based adaptive video streaming and generate datasets to build a QoE estimation model. The artifact is versatile and can be easily modified to accommodate additional ABS algorithms, cellular network traces, background traffics, video contents, etc. The complete artifact information for a reproducible purpose, including data generation, pre-process, and analysis code and datasets available online[2].

1.4 Thesis Outlines

This thesis is organized as follows: Chapter 2 presents the necessary background and discussion of related research works. Then, the design and implementation of `Edge QoE Probe` for QoE estimation and performance study are discussed in Chapter 3. Next, Chapter 4 provides the experimental evaluation analysis of the work. Finally, Chapter 5 concludes the thesis and discusses the limitations and directions of future work.

[2]https://github.com/sajibtariq/MSc_thesis

Chapter 2

Literature Review

This chapter provides the relevant background for our work and describes related work on similar issues and solutions to our approach.

2.1 Background

This section defines six main concepts in our research work: Quality of Experience (QoE) ecosystem focusing on Quality of Service (QoS), QoE assessment techniques, Probe as network traffic monitoring technique, HTTP Adaptive Streaming (HAS) standard video as most dominating internet traffic application, TCP and QUIC transport option for HAS and Multi-access Edge Computing (MEC) to take advantage of Edge Computing (EC) and Cloudification.

2.1.1 Quality of Experience (QoE) Ecosystem

It is expected that video traffic will reach 82% of the global IP traffic by the year 2022 [3] due to the advancement of digital multimedia technologies. Therefore, to meet user's satisfaction by providing excellent quality, it is necessary to monitor their satisfaction level from stakeholders' perspective, i.e., Content Providers, CDNs, and MNOs. Wherein for MNOs most common approach to measure user satisfaction relies on the QoS metrics/parameters collected from the network. The definition of QoS, according to the Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T), is:

"Quality of Service (QoS) is the totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service" [8].

Therefore, QoS is a network element's ability to assure that its traffic and service requirements would be satisfied. It is defined in terms of some parameters such as throughput/bandwidth, delay, packet loss. Specifically, QoS parameters describe the network system's technical performance or service without considering user perception and judgment. Consequently, the concept of QoE has developed as a user-centric reflection of QoS. The definition of QoE in the Qualinet white paper stated is:

"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" [24].

Hence, QoE is a user-perceived experience that may differ according to users' expectations and context. Figure 2.1 (a) illustrates the QoE ecosystem with important influence factors that directly or indirectly affect the user reaction/perception for any service. These influence factors

are co-related to each other. For video streaming, in most cases, application-level metrics such as video bit-rate, resolution, and buffering/stall have a strong influence on QoE [25].

QoS reflects the network and service level performance and is used to evaluate the quality of multimedia transmission (e.g., video streaming). Consequently, it directly affects application-level metrics, which later work as the most influential factor for changing user satisfaction or experience [26]. The relation between network-level QoS and application-level QoS (a.k.a objective QoE) is shown in Figure 2.1 (b). As a network operator, most influential factors are hardly measurable to predict QoE, but monitoring QoS metrics/parameters at the network level is viable and convenient. In terms of QoE and QoS, some metrics/parameters indicate the overall success of multimedia service (e.g., video streaming) known as Key Performance Indicators (KPIs). This thesis work focuses on network-level QoS KPIs (e.g., throughput, packet inter-arrival time and size) and application-level QoS KPIs, also referred to as objective QoE KPIs (e.g., resolution, bit-rate, stall). However, we use the QoE metrics term and QoE KPIs term interchangeably in some places.

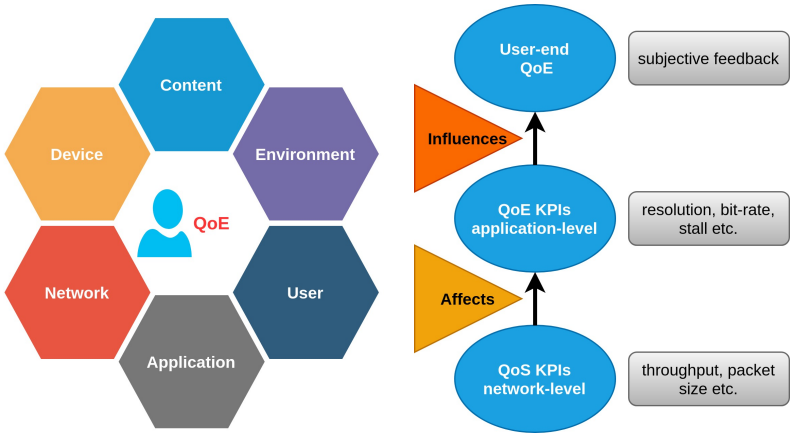


Figure 2.1: (a) QoE Ecosystem (b) Relation Between QoS and QoE

2.1.2 QoE Assessment Techniques

QoE is the level of user satisfaction or enjoyment with an application or a service. Therefore, it is essential for operators that users always stick with their service. For this reason, network operators require to assess the QoE in such a way they can react to the network quality degradation. The following subsections discuss the most common subjective and objective QoE assessment techniques and data-driven approaches for recent QoE assessment.

2.1.2.1 Subjective Assessment

Subjective quality assessment is a reference way to measure the quality by taking user feedback directly [27]. This assessment consists of several people participating as viewers of any particular sample multimedia service (e.g., video) and expressing their feelings or perception about that service. The Mean Opinion Score (MOS) is used as quantifying the users' feelings/perceptions. The MOS can take the values as shown in Table 2.1, which represent the corresponding degree of user satisfaction with the service [28]. Though subjective QoE assessment is considered the most accurate way, it suffers from a few significant drawbacks. *First*, this test is conducted

Table 2.1: Mean Opinion Score (MOS)

Values	1	2	3	4	5
Quality	Bad	Poor	Fair	Good	Excellent

in the laboratory environment, which should be unbiased, should not be affected by external noise, and should be close to the real-world scenario. Besides, it has a limited viewer of demography/diversity. *Second*, a high cost in terms of time, money, and manual effort. *Third*, quite impossible for real-time QoE evaluation.

These days crowd-sourcing is another promising direction considered an alternative for collecting subjective QoE ratings from users [29]. Crowd-sourcing connects to the internet beyond traditional laboratory environments and gives researchers a powerful tool to access a global pool of subjects. As a result, a diverse population and users' heterogeneity can be taken into account while simultaneously offering the possibility to extend laboratory studies (e.g., user-related influence factors or contextual factors). It is often not possible in a single test carried out in a test lab due to the restricted pool of subjects and limited contexts. However, one of the significant disadvantages of crowd-sourcing is the unreliability of the user rating. It is because incentives and payment schemes may influence users. Additionally, the test conditions and environments of the user are unknown in most cases. Thus, the impact of test conditions on the result is different from user to user and work to work.

2.1.2.2 Objective Assessment

Objective QoE assessment refers to the predict user behavior based on a mathematical model (e.g., expert model) instead of direct human judgment [30]. Such a model takes a set of objective input metrics/parameters. The mathematical model metric's output should correlate well (for validation) with the subjective test results, which serve as the ground truth QoE. Moreover, this assessment predicts quality automatically and performs fast enough. Therefore, service providers and network operators are mostly interested in this assessment technique.

As shown in Figure 2.2, we can categorize objective quality assessment techniques based on the source information they use. According to [31], we can define the Full Reference (FR), Reduce Reference (RR), and No Reference (NR) method depending on the availability of the source signal/information.

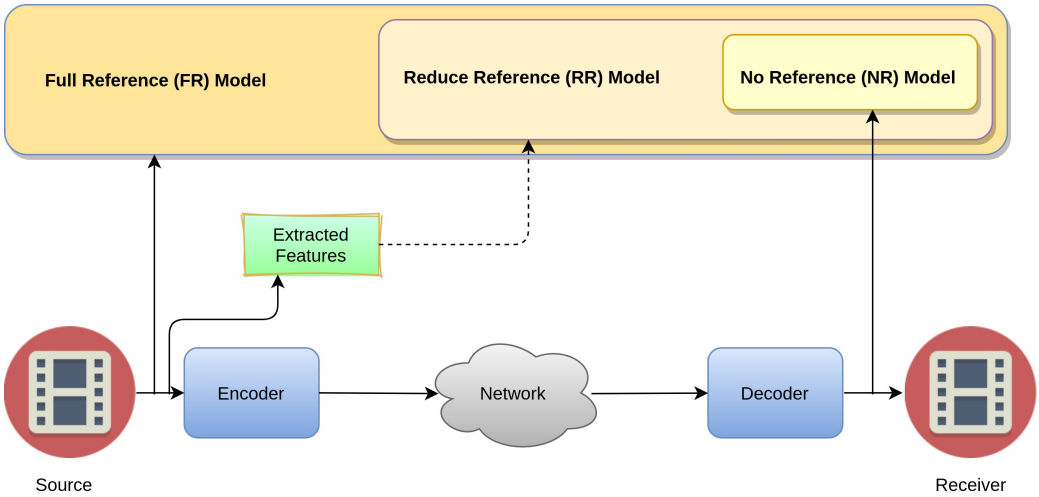


Figure 2.2: Objective Model Classification Based on Available Source Information

- **Full Reference (FR)**: This process compares the source signal and receiver signals to predict QoE (e.g., Peak Signal-to-Noise Ratio or PSNR, Structural Similarity Index Metric or SSIM). In this case, the information of the source signal is fully accessible.
- **Reduce Reference (RR)**: This process has partial access to the source signal. It predicts QoE by combining the received signal information with some extracted features of the source signal.
- **No Reference (NR)**: This process does not have any access to the source signal. It predicts QoE only based on the received signal.

In another approach [32], as shown in Figure 2.3, we can classify the objective quality assessment into five different groups according to the level of information used as an objective input parameter.

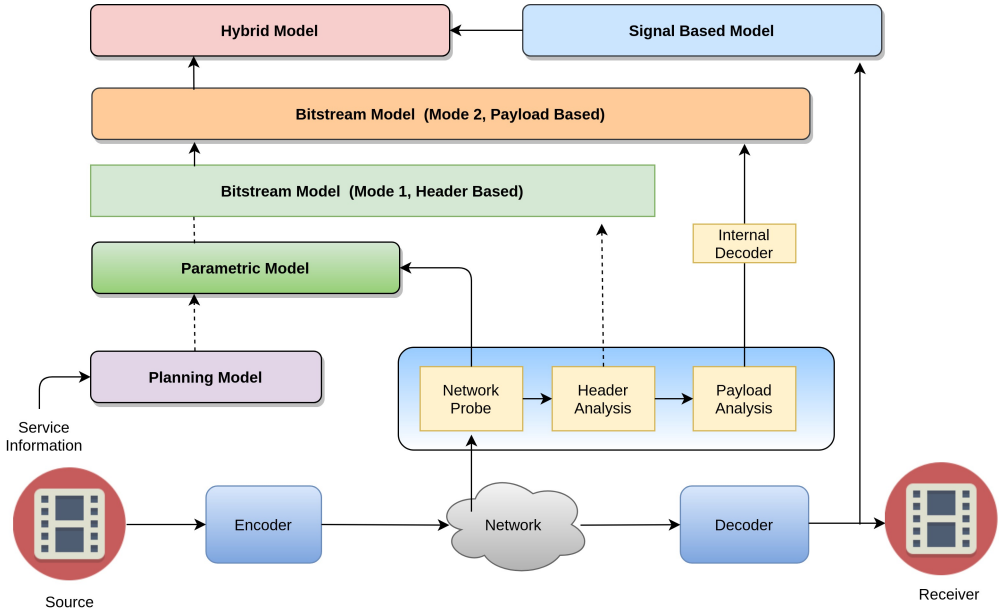


Figure 2.3: Objective Model Classification Based on the Information Used as Input Parameter

- **Media Layer Model (a.k.a Signal Based Model)**: This model uses the media signal (e.g., video) as an input parameter to predict QoE. FR and RR video quality methodologies fall into the media layer model.
- **Parametric Packet Layer Model (a.k.a Bitstream Header Based Model)**: This model uses IP/RTP packet header level information (e.g., throughput, packet loss, jitter) as an input parameter to predict QoE. In this case, the model does not need to look at payload information. Therefore, there is no requirement for media signal decoding information to predict quality.
- **Bitstream Layer Model (a.k.a Bitstream Payload Based Model)**: This model uses not only IP/RTP packet header level information but also payload information as an input parameter to predict QoE.
- **Hybrid Model**: This model uses the models mentioned above combinedly to predict QoE.

- **Planning Model**: This model takes assumed network and client parameters as input and predicts QoE. Besides, it usually requires prior knowledge about the system. Such models can be applied to network planning and terminal/application design.

2.1.2.3 Data Driven (ML-based) Assessment

Video streaming traffic over the internet using digital devices made a wide range of user context, choice/preference of contents, and system-related data available to the video content provides. Therefore, a data-driven approach emerges as a promising way for multimedia QoE evaluation. The data-driven approach's strategic decisions help predict more accurately user perspective QoE considering the subjectivity based on data analysis and interpretation. Since video content providers have direct access to user-related data and video quality from a user terminal, using those observed data, content providers can gain knowledge of the end-user QoE and an idea of video content choice preference. On the contrary, network operators (e.g., MNOs) do not have access to user-related data and only can track the traffic it passes through the network. The earlier section already stated limitations about the subjective assessment in terms of real-time QoE evaluation, and network operators are more prone to objectively QoE evaluation. Nevertheless, limitations still exist in the traditional objective assessment of QoE using the PSNR/SSIM when network traffic is encrypted. Due to the encryption, the visual quality of video and its variation can not be measured using PSNR/SSIM based objective assessment. Besides, DPI can not be read the quality of video directly from the network traffic payload. Prior work found QoS parameters from network traffic have a strong relation with QoE [26]. It is also feasible to predict QoE in terms of video quality metrics from classic QoS parameters [33] when traffic is encrypted. However, to make a correlation/mapping between QoS and QoE is not a trivial task. For this purpose, in recent ML strategy of achieving complex relationships between QoS and QoE has become a hot topic from both academic and industry perspectives. As the data-driven approach, rely on statistical and probabilistic as well as ML techniques, thus concluding that in-network QoE measurements are shifted to a data-driven approach.

Therefore, leveraging the data-driven approach, ML models are currently replacing the mathematical models (e.g., expert models) under the objective assessment to estimate the QoE. Wherein ML is a process of learning with a set of observations data in order to find any pattern in the data set and make better decisions for future data. ML usually follows four types of learning approach to make the ML model.

- **Supervised ML**: A set of observation data given in the input-output pair wherein the values are tagged with labels to identify the target output. This technique aims to make a function that defines the correlation between the inputs feature and target output. More specifically, in Supervised ML, a set of labeled data is given to training a model. Such a trained model later takes any new feature as input and predicts the target as output from experience. Supervised ML mainly divided into two categories.
  - **Classification**: Predict a category/discrete value as output (e.g., spam and not spam email). Algorithms are- Naive Bayes, Support Vector Machine, etc.
  - **Regression**: Predict a continuous value as output (e.g., house price). Algorithms are- Linear Regression, Ridge Regression, etc.
- **Unsupervised ML**: The observations of the data set present only the input values. More specifically, there is no target output exists as no values were tagged with labels. It aims to learn a function to find the pattern and similarities between values and make the

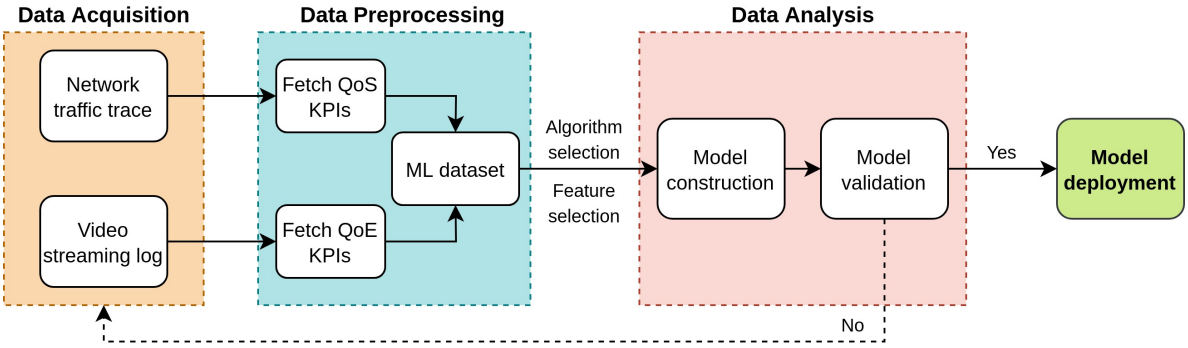


Figure 2.4: Supervised ML-based QoE Estimation Workflow

group of similar values in cluster form. Algorithms are- K-means, Principal Component Analysis, etc.

- **Semi-supervised ML**: It falls between supervised and unsupervised ML because it uses a data set to train a model with a small number of values tagged with a label and a large number of values without any tag.

- **Reinforcement ML**: This learning method allows a software agent to interact with its environment by taking action to maximize the notion of cumulative reward, where reward aids the agent in learning which action is best. Algorithms are- Markov Decision Process Q-learning, etc.

As we already know, network-level QoS parameters strongly impact application-level QoE, thus motivating using supervised ML to correlate network-level QoS parameters to QoE. For this purpose, a set of label data (tagging application-level objective QoE KPIs as target output and network-level QoS KPIs as feature input) must be built offline before training the supervised ML algorithms to make a QoE estimation model from network traffic measurement. Figure 2.4 presents a baseline workflow for applying supervised ML in the QoE estimation. A network operator can readily deploy such a trained model on its network premises and estimate the end-user perceived QoE of video streaming from real-time network traffic measurement. In this case, a monitoring agent named as a probe (described in the next section) is responsible for collecting network traffic measurements in QoS format. Due to encryption, such a probe only relies on the IP packet header; thus, we can place this QoE estimation in the *parametric packet layer model* objective QoE assessment category. Apart from this, there are multiple aspects of the probe scheme in the QoE evaluation solution, such as probe techniques, placement of the probe, monitoring probe type (e.g., physical or virtual).

2.1.3 Monitoring Probe Techniques

For network traffic monitoring and measuring, probes are used to extract and process information sent over the network — further, that information is used for quality assessment (e.g., QoE evaluation). The probes can be installed at any point in the delivery service delivery chain. As shown in Figure 2.5 (a) and (b), we can categorize probes into active and passive.

- **Active Probe**: Active probes send the extra test traffic over the network and then measure the network performance quality (e.g., Nmap[1]). Active probes do not examine

[1]https://nmap.org

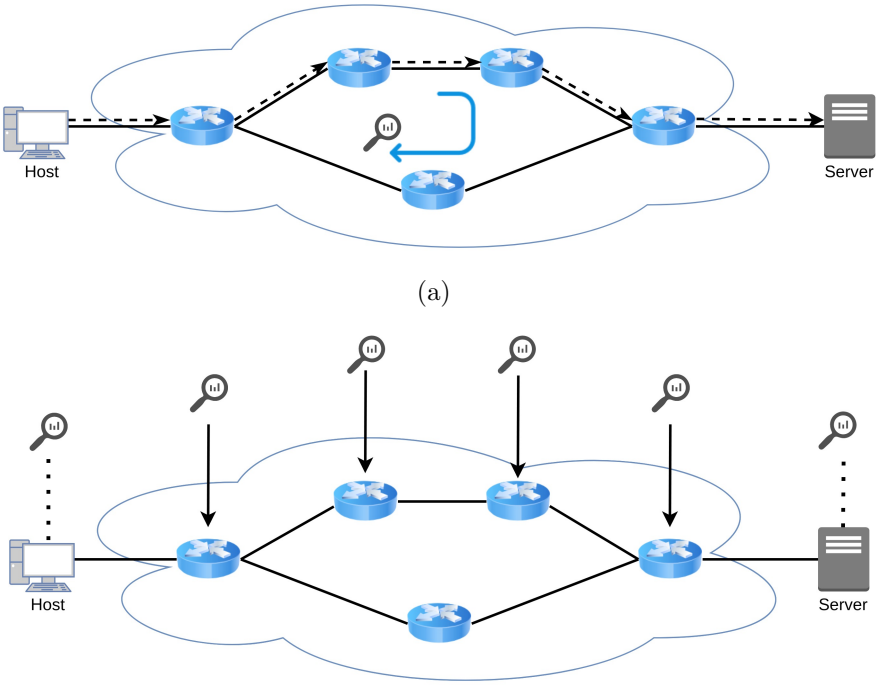


Figure 2.5: (a) Active Probe (b) Passive Probe

the actual user traffic, instead of it injects artificial traffic. Therefore, active probing has also known as synthetic monitoring. This monitoring helps to identify potential problem areas before affecting users. Active probes typically provide more detailed and reliable information from a user perspective. This type of probe grabs a small sample from an entire network scenario, so it is challenging to measure network quality's entire performance. The negative side of injecting extra test traffic over the network is to make congestion in the network system due to sharing the same network resources with real traffic, which may cause lousy network performance quality [34].

- **Passive Probe**: Passive probes passively inspect the network traffic that passes through the network (e.g., p0f[2]). A passive probe can be placed in any monitoring points such as server network, core network, access network (edge), or even user terminal based on preference. As passive probes pull real network traffic from any specific monitoring point, and unlike active traffic monitoring, it does not inject any additional data. Hence, passive probing does not interfere with the actual network traffic. It provides a holistic view of overall network performance quality. Moreover, passive probing helps network providers to identify the root cause that directly affects the user and traffic behavior pattern. Such traffic pattern later directly helps the QoE evaluation. DPI a traditional passive probing technique is used to capture the traffic passing through a monitoring point. DPI can quickly identify the network traffic type and monitors the objective QoE by extract KPIs from unencrypted payload data, but this can cause privacy and security issues. In contrast, for encrypted traffic, passive probing only can track and monitor network layer KPIs such as the number of packets, packet size, packets inter-arrival time, or throughput. Later leveraging ML techniques can classify the network traffic type and predict end-user

[2]https://github.com/p0f/p0f

QoE KPIs for specific service traffic flow. A passive probe can be traditional (physical) or virtual. The traditional probes are operated on physical devices, and it suffers from several limitations such as hardware dependence, lack of scalability, and flexibility. In contrast to a traditional probe, a virtual probe is a software or a function that works in a virtualized environment (e.g., VirtualBox[3] or Docker Container[4]). Besides, a virtual probe is more scalable and flexible than a traditional one because network operators can immediately deploy a new virtual probe in any location without requiring any dedicated hardware device. This makes virtual probe hardware independent and at a lower cost for network operators. The virtual probes are only viable for use when network infrastructures are virtualized; otherwise, traditional probes are used for traffic monitoring.

2.1.4 HTTP Adaptive Streaming (HAS) Standard

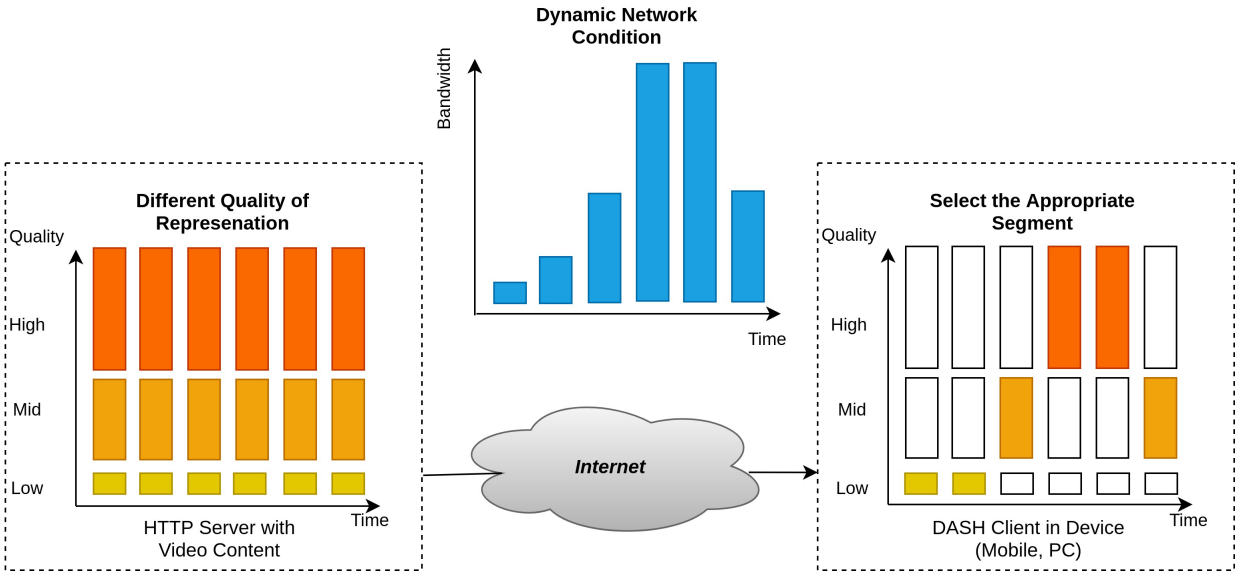


Figure 2.6: HTTP Adaptive Streaming (HAS) Overview

HAS has considered a de-facto standard for video streaming [7]. It is a combination of server and client and works by breaking the video content file into a sequence of small HTTP-based file segments (a.k.a chunks). Each segment contains a short interval (between 2 and 10 seconds in duration) playback time of a video with different representation level information (encoded with different bit-rates and resolutions). This information is comprised of an index file. Each different HAS implementation (e.g., Dynamic Adaptive Streaming over HTTP or DASH, HTTP Live Streaming or HLS) strategy has given different names of that index file. In DASH (a.k.a MPEG-DASH standard), the index file is called Media Presentation Description (MPD), an XML-compatible document. Over the internet, DASH is the most dominating format [18]. DASH format only specifies the structure of different representations of video content in MPD format. In a HAS streaming session, the DASH format compatible client first requests the MPD that contains the representation level information for video content. Based on that information, it asks the individual segments that include good video quality. In client-side Adaptive Bitrate Streaming (ABS) algorithms, they are mainly responsible for dynamically selecting the appropriate segments based on current network conditions (e.g., Bandwidth) and

[3]https://www.virtualbox.org
[4]https://www.docker.com

client playback buffer level. The purpose of this dynamic segment selection to adapt the network condition changes and avoid unwanted stall/re-buffering events. A representation of the HAS logic in dynamic network condition is presented in Figure 2.6. The ABS algorithms have three strategies to adopt the best segments as follows.

- **Rate-based**: This strategy estimates the Bandwidth based on delivery rates of previously downloaded segments and adopts the best representation (quality) of the next segment that fits this estimation [35].
- **Buffer-based**: This strategy monitors the state of playback buffer before every segment downloads and makes appropriate decisions for the next segment [36].
- **Hybrid**: This strategy is a combination of rate and buffer-based algorithms and adopts the best representation (quality) of the next segment from that two result [37].

2.1.5 Transport Options: TCP and QUIC for HAS Video Service

HAS was initially designed and implemented on top of TCP as the application-level HTTP standard requires a reliable transport protocol [19]. Afterward, HAS predominately used TCP long time for the benefits of reliability and in-order delivery. But, the initial HTTP/1.1 standard with persistent connection feature has suffered from a well-known Head of Line (HOL) blocking problem. Such a situation occurs as each client has limited TCP connections to the server and a delay in a new request queue over those connections. Although a pipeline feature was added later to make multiple requests over a single connection, the HOL problem has not been resolved because it requires responses to arrive in order. The next version, HTTP/2 [38] standardized by the IETF[5], has come with a multiplexing feature to overcome this issue. In HTTP/2 multiplexing, a single TCP connection can handle multiple requests in parallel, and responses do not require to arrive in order. It also embraces server push, stream priority, and stream termination features. Nevertheless, another kind of HOL blocking still exists in TCP transport for HTTP/2 standard. When HTTP/2 uses TCP, if a packet loss occurs in the TCP stream, it makes all subsequent TCP streams wait until that packet is retransmitted and recovered.

Due to the data privacy and security issues in recent OTT platforms are delivering their streaming services with encryption. A new Transport Layer Security (TLS) is imposed over the TCP and under the HTTP to facilitate the privacy and data security for communications over the internet through encryption. In this work, we adopt the TCP term considering the combination of HTTP+TLS+TCP, which is also referred to as HyperText Transfer Protocol Secure (HTTPs). TLS requires a new handshake to ensure that the session is secured alongside the initial TCP handshake, leading to a little time-consuming connection establishment. In HAS, connection delay or retransmission due to HOL issue and handshake latency due to TLS and TCP may cause unwanted delays while downloading video segments and force the ABS algorithm to adopt degraded quality segments..

The shortcomings mentioned above of TCP lead to the development of alternative transport protocol such as Google developed QUIC protocol running on top of UDP [21]. QUIC inherits all the HTTP features over TCP, aiming to reduce connection establishment, improving congestion control, multiplexed/pipelined requests without HOL blocking, forward error correction, and seamless connection migration. Recently, QUIC was adopted as a transport protocol for the

[5]https://bit.ly/3qORHbk: The Internet Engineering Task Force (IETF) is the leading Internet standards body. It develops open standards through open processes with one goal in mind: to make the Internet work better.

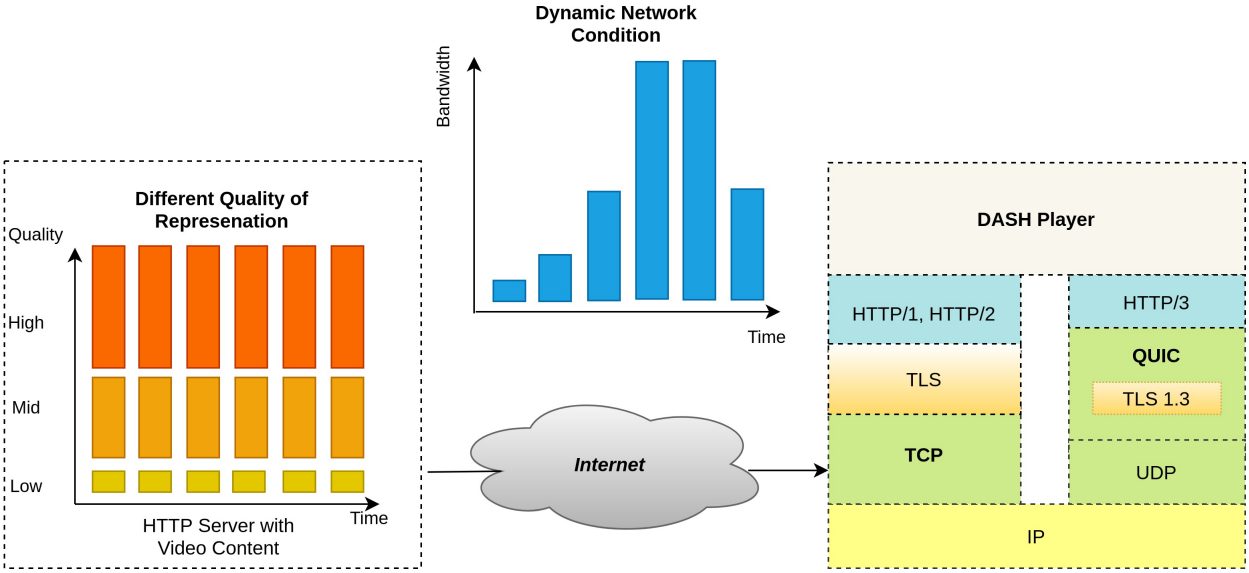


Figure 2.7: DASH Player over TCP and QUIC to Download the Video Segments

HTTP/3 standard [16]. Therefore, the HTTP/3 standard over QUIC uses a single handshake for a secure connection and avoids the HOL blocking issue from the multiplexing feature. In HTTP/3 over QUIC multiplexing, each stream is independent of the other, and subsequent streams are not affected while a particular stream packet loss occurs. Figure 2.7 depicts how the client-side DASH player works with TCP and QUIC transport.

2.1.6 Edge Computing (EC) and Cloudification

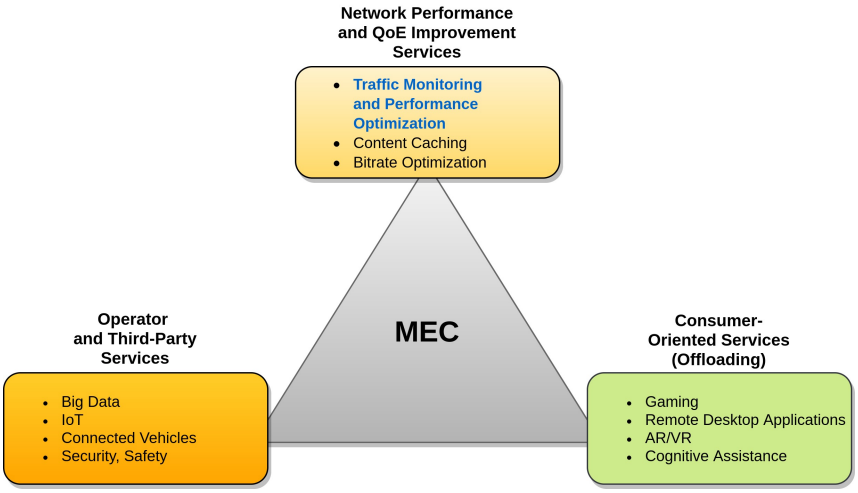


Figure 2.8: MEC Use Cases

DASH clients are allowed to switch between different video qualities to improve the viewing experience, matching based on current network conditions. Thus, there is a need to be aware of network conditions to optimize video delivery. A new QoE-centric cutting-edge technology, EC, optimizes video delivery and meets user demand. It allows pushing applications, data, and computing power to the edge of the operator's network. The ETSI promotes Multi-access Edge Computing (MEC) [39] a virtual platform paradigm to implement the EC concept. MEC provides an IT service environment and cloud-computing capabilities at the mobile network's

edge, integrating the Radio Access Network (RAN). Besides, it also offers a service environment with low latency and high bandwidth. Figure 2.8 depicts the potential use cases of MEC [40]. We shed light on the *"Network Performance and QoE Improvement Services"* category among all the possible use cases. More specifically, network traffic monitoring and network (e.g., backhaul/fronthaul) performance optimization under this category.

Regarding this, a MEC *monitoring network function* will provide the real-time network-level (e.g., backhaul/fronthaul) traffic information to a MEC *analytic network function*, which role is to compute the traffic requirements if any degradation occurs at the backhaul (e.g., Core, Internet) or fronthaul (e.g., RAN) end. Later, a MEC *optimization function* will optimize the network according to traffic requirements calculated by the analytic function. Such a use case can also help to make co-ordination between the backhaul and fronthaul network. The optimization function can optimize the network in several ways, such as-

- **Backhaul**: Traffic re-routing (reactive).
- **Backhaul**: Reshaping the traffic per application (reactive/proactive).
- **Fronthaul**: Increase/reduce the power of microwave links based on actual capacity need (reactive/proactive).

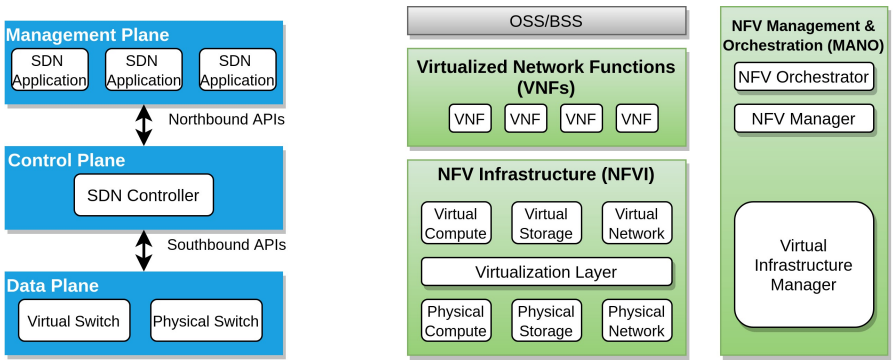


Figure 2.9: (a) SDN Architecture (b) NFV Architecture

However, MEC virtual platform can cope with the future network (e.g., 5G and Beyond) cloudification via Network Functions Virtualization (NFV) [11] and Software-Defined Networking (SDN) [10]. While NFV serves as a technology for decoupling network functionality as a Virtual Network Function (VNF) from the hardware appliance. Figure 2.9 (b) presents the basic NFV architecture composed of three key elements: Network Function Virtualization Infrastructure (NFVI), VNFs, and NFV Management and Orchestration (MANO). Specifically, a VNF would be deployed on a virtual resource (e.g., NFVI) in the form of a Virtual Machine/Docker Container with a specific specification that operates a piece of a software implementation of network function. And, NFV MANO will be responsible for taking adequate actions to manage all virtual-specific tasks in the NFV framework.

In contrast, SDN separates the control and forwarding/data planes, and logically centralizes network intelligence/brain into the SDN controller. SDN controller has a global view of the underlying network (e.g., NVFI or physical) and enables the underlying networking infrastructure's programmability. Figure 2.9 (a) describes the architecture of SDN. The control plane consists of an SDN controller with a software-based service called network operating system (NOS) and can contain network and control applications. The data plane includes the

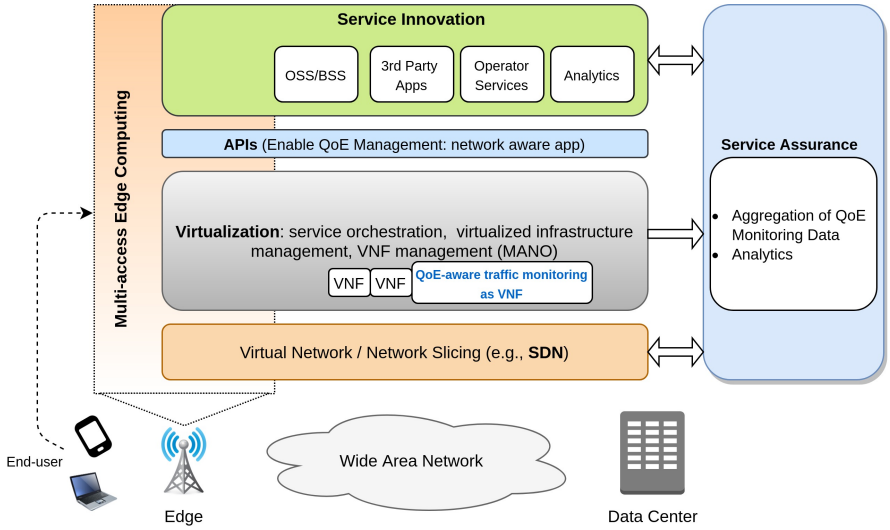


Figure 2.10: Future Softwarized Network for QoE Monitoring and Management (adopted from [9])

forwarding device (e.g., virtual or physical switches). The southbound interface defines the communication protocol (e.g., OpenFlow) between the data plane's forwarding devices and the control plane. On the other hand, the northbound API opens a common interface between the management plane and the control plane. Wherein the management plane contains a set of SDN applications, such as QoE-aware re-routing, QoE-aware network optimization. In a nutshell, a management plane defines the high-level policies (e.g., a set of rules), which are ultimately translated via the SDN controller to southbound-specific instructions that enable the forwarding devices' programmability.

The MEC ecosystem can leverage the NFV concept to obtain virtualization and SDN to make the underlying network infrastructure programmable. To the end, MEC as a virtual platform extends the notion of the SDN's QoE monitoring and management approach at the network edge where a passive probe for QoE-centric network traffic monitoring would work as a VNF. Figure 2.10 depicts the high-level scenario of future network scenarios, including the most recent technological QoE management approaches. A service assurance block will gather monitoring data for analytics and inform the edge controller for taking concrete action to QoE-centric network optimization.

This work showcases a lightweight technique of passive network traffic monitoring for DASH-supported video service at the network edge and trains the supervised ML algorithms for user-level QoE metrics (KPIs) estimation. Evaluation of such QoE-centric monitoring and estimation information will help network orchestrators (e.g., edge SDN controller) to overcome network degradation. The purpose of this work is to monitor the network-level traffic information and estimate the end-user QoE, not about how the network takes concrete action to overcome QoE drop.

2.2 Related Work

This section covers QoE measurement approaches and QoE performance evaluation over traditional TCP (HTTPs) and QUIC transport for DASH video services. A fair amount of diverse methods are already stated for end-user QoE estimation. Figure 2.11 shows a complete hierarchy of related work in the area of QoE measurement based on the QoE monitoring data

and implemented strategies. This section is organized into the following category. The first category discusses client-level QoE measurement. The QoE measurement approaches at the network-level for unencrypted and encrypted traffic are presented in the second category. Besides, the most recent techniques using ML for QoE estimation are elaborately presented under this category. The third category presents a hybrid approach for QoE estimation. In the fourth category, we showcase related work for the QoE-centric strategy in EC. Apart from this, a brief overview of DASH video performance evaluation works over TCP and QUIC is shown in the fifth category. Finally, we provide a brief comparison of this thesis work's contribution in contrast to all state-of-the-art QoE measurement approaches.

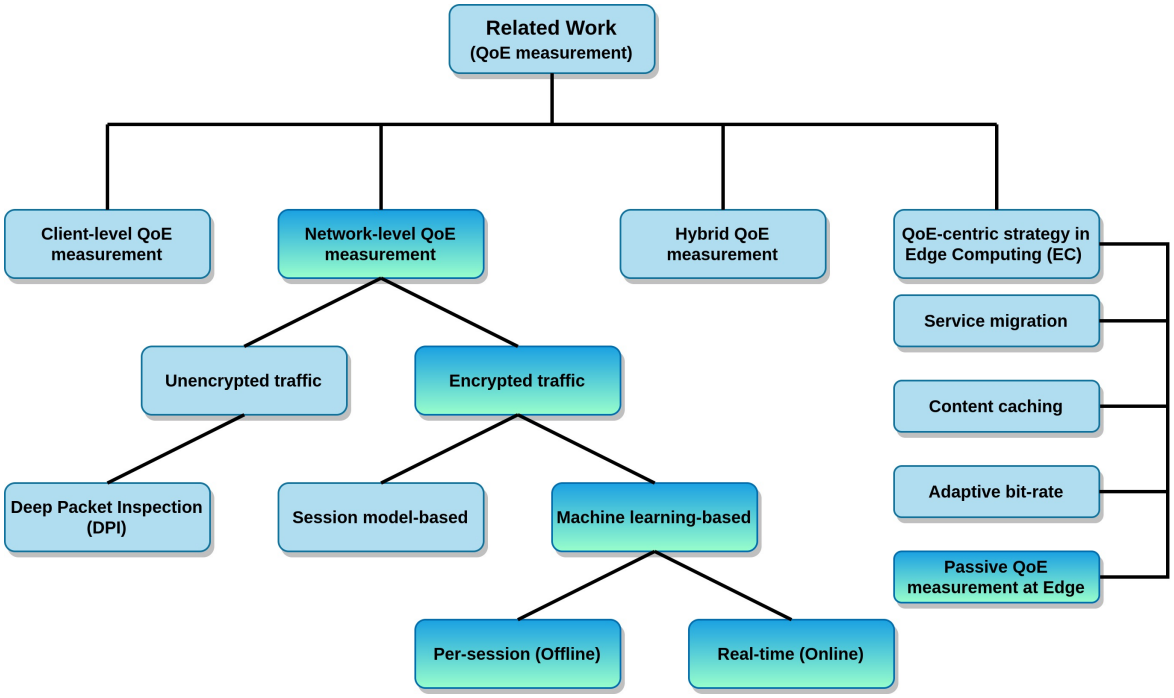


Figure 2.11: Related Work Classification in the Area of QoE Measurement

2.2.1 Client-level Measurements

Many approaches are based on assessing QoE from direct end-user devices. A tool (e.g., application or software) is installed on the client-side can passively grab the user device's capabilities information, user context information, user content usage preference, content's application-level metrics, and network statistics. Later, such information helps to evaluate end-user QoE at the application level.

Work [12] [41] [42] [43] [44] [13] are based on end-user cooperation to install a dedicated tool for evaluating the video streaming QoE. Wamser et al. in [12] [41] developed an Android app YoMoApp (YouTube Performance Monitoring Application) to passively monitors KPIs (e.g., player state/events, buffer, and video quality level) of YouTube on end-user smartphones. Also, it supports collecting subjective QoE feedback from end-user along with network usage statistics and device characteristics. Similar to YoMoApp, Nam et al. developed YouSlow in [42], a web browser plug-in that can monitor YouTube stalling events while clients watch YouTube videos on Chrome browsers. The monitoring data then marked on Google maps with Internet Service Providers (ISPs) statistics for stall event duration and approximate location. Joumblatt et al. in [43] proposed a passive measurement of collecting network performance and

direct user feedback by a tool called HostView installed in the end-user devices to infer QoE. Similar kinds of approaches used Chen et al. in [44] to capture the user experience over a network application by a framework called OneClick run in the user devices. Users were asked to click a dedicated button whenever they feel dissatisfied with the application's quality. Another tool named Pytomo, Juluri et al. in [13] designed to analyze the YouTube videos for evaluating user QoE. This tool emulated the user behavior by downloading and playing a YouTube video and collected statistics of the video download and playback (e.g., initial buffer, interruptions, total buffering duration, buffer duration at the end of the download).

2.2.2 Network-level Measurements

However, end-user provided application-level QoE metrics are not accessible by network operators (e.g., MNOs). They do not have any access to the video streaming app or on the user's device to install any probe mechanism (e.g., tool). For this reason, in the second category, the operator relies on in-network measurement to monitor the application-level metrics as well as end-user QoE.

2.2.2.1 Unencrypted Traffic

DPI. For a long time, network operators rely on DPI, a passive probing in most cases for unencrypted traffic. DPI helps to extract the application-level information directly from the network traffic payload. In video streaming service (e.g., DASH), using DPI operators can instantly recognize the HTTP GET request and response message for each video segment/chunk to infer video QoE. Work [14] [45] [46] [47] [48] [49] [15] are based on information obtained from network-level by DPI without user cooperation to infer the QoE. Schatz et al. in [14] showed the YouTube QoE monitoring approaches for ISPs by the passive probe. According to their proposed method, it is feasible to detect application-level stalling events at high accuracy by using network-level DPI. Mangla et al. in [45] developed MIMIC, a passive network measurement system to estimate the QoE metrics such as average bit-rate, re-buffering ratio, and bit-rate switches of video sessions by investigating the request-response pattern of HTTP logs. Later, Mangla et al. in [46] proposed VideoNOC, a prototype to assess adaptive video QoE metrics using the passive measurement for MNO. For this purpose, they collected HTTP/HTTPs data using a web proxy. They also processed them in a distributed manner to estimate QoE metrics such as video quality, re-buffering ratio, and quality switches. Farshad et al. in [47] proposed an in-network QoE monitoring framework on an SDN-based network where video streaming related traffic was replicated toward the measurement agent to parse the manifest file (HTTP GET request and response message) for objective QoE predicting. Huysegem et al. in [48] proposed 'HAS probe' a session re-construction (containing parsed HTTP GET request and response sequence through DPI) framework based packet capturing from network element without requiring client level monitoring to infer the QoE. Wu et al. in [49] proposed an ML-assisted approach based on the segment retrievals log information from the CDN node to detect the freeze (stall) events of the DASH client. Krishnamoorthi et al. in [15] proposed BUFFEST, a buffer state emulator to inferring video streaming re-buffering, leveraging DPI on the content of HTTP requests in the clear or on unencrypted HTTPs requests by a trusted proxy.

2.2.2.2 Encrypted Traffic

Nowadays, the widespread use of encryption in OTT provided video streaming traffic made network operators unable to infer the application-level QoE metrics' by the DPI approach. Hence, end-to-end encryption makes the in-network measurement more challenging to network operators for figuring QoE. There are two approaches operators uses to tackle this challenge: *1. Session Modeling-based (SM)* and *2. Machine Learning-based (ML).*

Session modeling-based. SM approach requires the properties of underlying streaming protocols to infer QoE by modeling a video session. For unencrypted traffic, previously in [45] [48] work presented video sessions reconstruction by extracting directly the video segment/chunk request-response pattern of HTTP through DPI. In [50], the authors presented the eMIMIC extension of MIMIC [45] a video session modeling to infer QoE metrics, such as average bitrate and re-buffering ratio from encrypted traffic. In this work, the authors reconstructed the HTTP segments/chunk delivery sequence of the video session based on packet header information. Besides, they compared the performance of eMIMIC with ML-based solutions and highlighted eMIMIC obtained better performance. Nevertheless, eMIMIC suffers from two drawbacks- firstly, the video session reconstruction was based on HTTPs traffic. Thus, such a model is not applicable for the QUIC transport protocol, which supports multiplexing features. Next, eMIMIC was based on considering fixed-length video segment/chunk, thus makes it impracticable for video streaming service, which uses variable-length video segment/chunk (e.g., YouTube). In the end, the eMIMIC can lead to error in session reconstruction for new underlying streaming protocols because this solution was based on specific services and protocols.

Machine learning-based. The diversity of the devices, streaming services, network types and protocols has made the SM-based solution more complex. For network operators, such a solution is not flexible enough to adapt if the content provider changes the adaptation strategy. Due to SM-based solutions' inflexibility, recent studies leverage ML-based approaches to estimate QoE metrics/KPIs from encrypted traffic. ML approach infers the application-level objective QoE KPIs by correlating network-level QoS KPIs (e,g, throughput, loss, delay). Specifically, supervised ML uses a significant number of network-level QoS KPIs as features and ground truth (e.g., application-level QoE KPIs) as targets to build a correlation model. For video streaming, the supervised ML-based approach is divided into two methods. They are *Per-session* and *Real-time*. In the per-session KPIs estimation approach (a.k.a Offline), the entire video session generated network-level QoS KPIs features are used to make a classification/regression-based prediction model. On the contrary, in the real-time KPIs estimation approach (a.k.a Online), network-level QoS KPIs features generated on a specific time window in video session are used to make a prediction model calculating QoE KPIs on that time window. A brief overview of in-network per-session and real-time ML-based QoE estimation related works for adaptive video streaming is given in the following Table 2.2.

2.2.3 Hybrid Measurements

The next category for QoE measurement relies on a hybrid approach, a combination of client and network side measurement to estimate the user-end QoE from the network operator perspective. In such methods, the client must provide specific information to the network to estimate QoE [61]. However, this approach is still needed user cooperation and painstaking to use.

Table 2.2: ML-based Per-session and Real-time QoE KPIs Estimation Approaches

Approach	Work	Target QoE	Methodology
Per-session (a.k.a Offline)	Dimopoulos et al. in [33]	Per-session KPIs classification (stall, average quality and quality variation)	Streaming Service- YouTube Transport- TCP Features- Network and transport level traffic Ground Truth- Collected by web proxy
	Orsolic et al. in [51]	Per-session QoE performance classification	Streaming Service- YouTube on android Transport- TCP Features- Network level traffic Ground Truth- Collected by IFrame based YouQ app
	Orsolic et al. in [52]	Per-session MOS (ITU-T P1.203) and KPIs classification (resolution, stall, initial delay and bit-rate)	Streaming Service- YouTube on iOS platform Transport- TCP Features- Network level traffic Ground Truth- Collected by *stats for nerds*
	Vasilev et al. in [53]	Per-session KPI classification (re-buffering ratio)	Streaming Service- AMuSt simulation platform Transport- TCP Features- Network and transport level traffic Ground Truth- AMuSt
	Khokhar et al. in [54]	Per-session MOS (ITU-T P1.203) and KPI (startup delay) regression, and KPIs (quality, stall and quality switches) classification	Streaming Service- YouTube Transport- TCP Features-Network and application level traffic Ground Truth- Collected by YouTube data API
Real-time (a.k.a Online)	Mazhar and Shafiq in [55]	Real-time KPIs classification (initial delay, stalling and resolution)	Streaming Service- YouTube in browser Transport- TCP and QUIC Features- Network and transport level traffic Ground Truth- Collected by YouTube IFrame API ***Prediction Granularity- 10 second***
	Gutterman et al. in [56]	Real-time KPIs classification (buffer warning, streaming phase and resolution)	Streaming Service- YouTube in browser Transport- TCP and QUIC Features- Application level traffic (with segment detection) Ground Truth- Collected by YouTube IFrame API ***Prediction Granularity- 5 second***
	Schmitt et al. in [57]	Real-time KPIs classification and regression (resolution and initial delay)	Streaming Service- Netflix, YouTube, Amazon and Twitch Transport- TCP and QUIC Features-Network, transport and application level feature (with segment detection) Ground Truth- Collected by Chrome extension ***Prediction Granularity- 10 second***
	Seufert et al. in [58]	Real-time KPI classification (stall event) **Enable entire session KPIs estimation**[a]	Streaming Service- YouTube in browser Transport- TCP and QUIC Features- Network level feature (**temporal feature**[b]) Ground Truth- Collected by JavaScript-based monitoring script ***Prediction Granularity- 1 second***
	Wassermann et al. in [59]	Real-time KPI classification (resolution)	Streaming Service- YouTube in browser Transport- TCP and QUIC Features- Network level feature (**temporal feature**[b]) Ground Truth- Collected by JavaScript-based monitoring script ***Prediction Granularity- 1 second***
	Orsolic and Skorin-Kapov in [60]	Real-time KPIs classification (resolution and bit-rate) **Enable entire session KPIs estimation**[a]	Streaming Service- YouTube in android and iOS platform Transport- TCP Features- Network level feature (**temporal feature**[b]) Ground Truth- Collected by *stats for nerds* ***Prediction Granularity- 1 second***

[a] Aggregating real-time temporal QoS features for all time window can evaluate the entire video session QoE KPIs

[b] A series of QoS features computed over a time period (e.g., multiple time window)

2.2.4 QoE-centric Strategy in EC

Apart from the QoE measurement approached above-mentioned, EC's QoE-centric strategy is the trending topic these days. Several works presented a QoE-centric scheme for EC concepts in terms of DASH video service leveraging the MEC paradigm. Most of the EC's work mainly focuses on service migration, content caching, and adaptive bit-rate use cases.

Service migration. The authors in [62] discussed the impacts and benefits associated with network service migration from the cloud to fog nodes for video distribution with QoE support. Dinh-Xuan et al. in [63] proposed a VNF for video traffic monitoring in the network. They evaluated its accuracy depending on different placements (network edge vs. close the streaming server).

Content caching. Xu et al. in [64] proposed a MEC enhanced video delivery scheme combining content caching and streaming technology together where the MEC server acts as a controlling component to implement the video caching strategy and adjust the transmitted videos bit-rate flexibly. Ge et al. in [65] presented a content caching framework to improve user QoE at EC for DASH video streaming applications. In [66], the authors presented a QoE-driven mobile-edge caching placement optimization strategy for dynamic adaptive video

streaming. Liang et al. in [67] presented QoE-aware wireless edge caching with bandwidth provisioning and caching strategies in SDWNs to decrease the content delivery latency and improve the utilization of the network resources. Behravesh et al. in [68] showed an ML-driven predictive prefetch and caching approach for DASH content in MEC-enable mobile networks.

Adaptive bit-rate. Li et al. in [69] proposed a novel architecture for adaptive video streaming, including an adaptation algorithm running as a MEC service, aiming to relax network congestion while improving user experience. Kim and Chung in [70] presented an EC-assisted greedy based bit-rate allocation algorithm that jointly optimizes the QoE, resources, and fairness among the DASH clients.

Passive QoE measurements at EC. In the scope of QoE monitoring and measurement at the edge, there was no such significant work has not been done to date. A demonstration of real-time QoE estimation of DASH-based mobile video applications through EC is presented in [71]. The authors implemented a virtual function in LTE network edge premises integrating a proxy (e.g., DPI), which plays a role in breaking the encryption and reading the QoE information from packet headers to evaluate real-time streaming quality. Their proposed algorithm computed per 0.5 second's video buffer length with other QoE metrics (e.g., initial delay, playback duration, and re-buffering duration). On the other hand, Zheng et al. in [72] proposed a Fog-assisted Real-time QoE Prediction (FRQP) scheme to enable service providers to estimate DASH video QoE at fog (edge) nodes. The authors showed a passive probing technique at fog nodes with the supervised ML model's help to infer users' QoE by observing network traffic packet headers. For this, the authors proposed a heuristic approach to split the playback buffer status into two sub-phases (e.g., re-buffering or no re-buffering) based on the ON-OFF traffic pattern between the user and the DASH server. They stated a time sequence-based temporal technique to collect per time window real-time QoS features, i.e., download throughput, moving average download throughput, and request distance (density of request packets). However, the authors did not precisely mention the granularity of the time window to evaluate QoE prediction (e.g., re-buffering or no re-buffering) from QoS features.

Apart from work in the aforementioned EC zone, for the virtualization and softwarization context, related work in [73] discussed the deployment of virtualized monitoring probes considering the 5G network scenario at MEC platforms. Furthermore, work in [32] [74] [9] provided challenges and a detailed overview of QoE-oriented MEC architecture in the context of SDN and NFV.

2.2.5 QoE Performance Evaluation over TCP and QUIC

Table 2.3 presents relevant details about the work related to the performance of adaptive video streaming over TCP and QUIC transport.

2.2.6 This Work vs. State-of-the-Art

In this work, we propose a lightweight, fine-grain passive encrypted network traffic monitoring technique from an edge node for real-time DASH video service QoE estimation. We adopt an approach similar to the most recent three works [58] [59] [60] as stated in the Table-2.2 for ML-based real-time QoE estimation. Since a stall or re-buffering event is rarely observed in the shortest granularity time window, this work emphasizes displayed video quality such as resolution and bit-rate QoE metrics. However, this work is based on an emulation-based DASH

Table 2.3: Related Work in the Area of DASH Video Performance Evaluation over TCP and QUIC (Category 1 and 2) and Current Research Approaches for Getting Better DASH Video Performance over Newly Standardized QUIC Transport (Category 3).

Category	Work	Main Insight
1	Timmerer and Bertoni in [75] evaluated TCP and QUIC on dynamic adaptive streaming with varying network link utilization and throughput. The authors stated that QUIC does not provide improvement in the overall streaming performance. Bhat et al. in [76] evaluated the performance of the ABS algorithms by head-to-head comparing between TCP and QUIC transport. The authors stated that QUIC does not provide significant QoE benefits to the existing ABS algorithms because these algorithms were designed over TCP.	**(1) QUIC offers poor performance than TCP for adaptive video streaming. (2) An open-source implementation is applied for QUIC transport.**
2	Arisu and Begen in [77] evaluated QUIC in terms of the users' frame-seek requests and frequent network changes. They used Google provided toy server and client (player) and showed QUIC provided better QoE by reducing the wait times and the buffer starvation rates. Kakhki et al. in [78] found that QUIC provided better streaming QoE, but only for high-quality video-streaming using YouTube. Zinner et al. in [79] showed QUIC with 0-RTT connection establishment performed better than the other protocols for the playback start in YouTube video streaming.	**(1) QUIC offers better performance than TCP for adaptive video streaming with Google provided QUIC server and player. (2) In category 1, QUIC was less competitive than TCP due to using open source implementation of QUIC or not using Google provided server with the latest version of QUIC transport.**
3	Li et al. in [80] and Hayes et al. in [81] provided MMT and SDN based approach to improve the QoE of adaptive streaming over QUIC transport. Bhat et al. in [82] showed QUIC performed better than TCP with a modified DASH-based ABS approach (SQUAD), which inherits retransmit segments' ability to improve to QoE of a viewer watching the video. Nguyen et al. in [83] presented a retransmission technique (H2BR) for the modified ABS algorithms perform well by improving the average video quality with HTTP/3 atop QUIC compared to HTTP/2 atop TCP in the context of packet loss and retransmission.	**(1) Novel strategies to make the QoE performance of adaptive video streaming more robust with QUIC transport. (2) Modification of ABS algorithms to obtain better performance over open-source implementation of QUIC transport.**

video service, which allows the use of different adaptation logic for video segment selection instead of a particular streaming service (e.g., YouTube) adaptation logic. A brief comparison between our work and state-of-the-art most recent three works is presented in Table 2.4.

As our work focuses on QoE estimation at the edge premises, we also provide a brief comparison (Table 2.5) between our work and the works [71] [72] carried out in EC's scope stated in the earlier subsection 2.2.4 for QoE measurements.

Lastly, in this work, we conduct all the experiments using traditional TCP and open-source implementation of QUIC transport (similar to category 1 stated in Table 2.3) for QoE estimation and QoE performance evaluation.

2.3 Literature Review Summary

Quality of Experience (QoE) is a user-centric experience that depends on numerous factors and is assessed both subjectively and objectively. Wherein Machine Learning (ML)-assisted

Table 2.4: A Comparison Between This Work and ML-based Real-time QoE Estimation State-of-the-Art Works

		Work [58]	Work [59]	Work [60]	This Work
Target QoE	**Real-time**	Stall Event (Binary Classification)	Resolution (Multi-class Classification)	Resolution (Binary Classification) Bit-rate (Binary Classification)	Resolution (Multi-class Classification) Bit-rate (Multi-class Classification)
	Per-session	Startup (initial) Delay (Regression) Stall Event Number (Regression) Stall Event Duration (Regression) Stall Ratio (Regression)	-	Resolution (Binary Classification) Bit-rate (Binary Classification) ITU-T MOS (Multi-class Classification)	Resolution (Multi-class Classification) Bit-rate (Multi-class Classification) ITU-T MOS (Multi-class Classification/Regression) Quality Switches (Multi-class Classification) Stall Ratio (Multi-class Classification) Startup Delay (Regression)
Granularity	-	1 second	1 second	1 second	0.5 second
Temporal Time Window	-	Current Trend Session	Current Trend Session	Current Trend -	Current Trend Session
Basic Features	-	Packet Size Packet Count Packet IAT[a] -	Packet Size Packet Count Packet IAT[a] -	Packet Size Packet Count Packet IAT[a] Throughput	Packet Size Packet Count Packet IAT[a] Throughput
Total Features	**Real-time**	208 (TCP and QUIC)	208 (TCP and QUIC)	218 (TCP)	252 (TCP) 168 (QUIC)
	Per-session	208 (TCP and QUIC)	208 (TCP and QUIC)	62 (TCP)	216 (TCP) 144 (QUIC)
Window Used for Session QoS Aggregation	-	All (Current, Trend and Session)	All (Current, Trend and Session)	All (Current and Trend)	Current
Streaming Service	-	YouTube	YouTube	YouTube	Emulation-based DASH

[a]Inter Arrival Time

Table 2.5: A Comparison Between This Work and QoE Estimation Works in the Scope of EC

	Work [71]	Work [72]	This Work
QoE Measurement at Edge	✓	✓	✓
DPI	✓	✗	✗
ML-based	✗	✓	✓
ML Model Benchmark	✗	✗	✓
Real-Time Target QoE	Buffer Length	Buffer Status	Resolution and Bit-rate
Prediction Granularity	0.5 second	Vague	0.5 second
Basic Features	-	Throughput and Request Density	Throughput, Packet Size/Count/IAT[a]
Transport Option	TCP	Vague	TCP and QUIC

[a]Inter Arrival Time

passive probing-based parametric objective QoE assessment is the most widely used technique nowadays. The network-level Quality of Service (QoS) metrics play the most vital role in altering the QoE for the most dominating Dynamic Adaptive Streaming over HTTP (DASH) video service. Besides, DASH video service works based on adapting network function status, and Over-the-Top (OTT) platforms deliver the DASH service to end-user by end-to-end encryption over TCP (HTTPs) and QUIC transport. On the other hand, the concept of Edge Computing (EC) facilitates the network operator to keep aware of network conditions and optimize video delivery of such DASH service by passively monitoring QoS metrics and assessing QoE. Several approaches were applied to QoE measurements in the literature, such as client-level, network-level for encrypted and unencrypted traffic, hybrid, and edge-centric measurement. Moreover, a couple of works were carried out for QoE performance evaluation over TCP and QUIC transport. We provided a brief comparison of this thesis work with the most recent related work for QoE measurement and performance evaluation.

Chapter 3

Edge QoE Probe Design and Implementation

This chapter presents the approach used in the design and implementation of the `Edge QoE Probe` to passively monitor the traffic containing video service information at the network's edge premises for parametric ML-based QoE estimation. Besides, we discuss the impact of different transport protocols on DASH video service performance.

3.1 Design Overview

For our proposed predictive `Edge QoE Probe` scheme Figure 3.1 shows a high-level network architecture compound with end-user, access network (AN), which can be a wireless network such as WiFi and LTE, core network, internet, and video source. The location of the video source can be located anywhere on the internet or in-network operator premises. The packet flow from/to video source passes through the internet, core, and access. We deploy a predictive QoE probe mechanism for DASH video service at the edge of the network where AN is considered an edge node. Such a predictive `Edge QoE Probe` can act as a VNF for a MEC facility located in edge premises. Network operators are mainly responsible for managing the edge premises (e.g., AN) and backhaul network (e.g., core). From the network operators' point of view, the network's edge is considered closer to the user premises to observe the nearly accurate end-user video service's performance. Also, the monitoring traffic at the edge location is less mixed with cross-traffic than other locations.

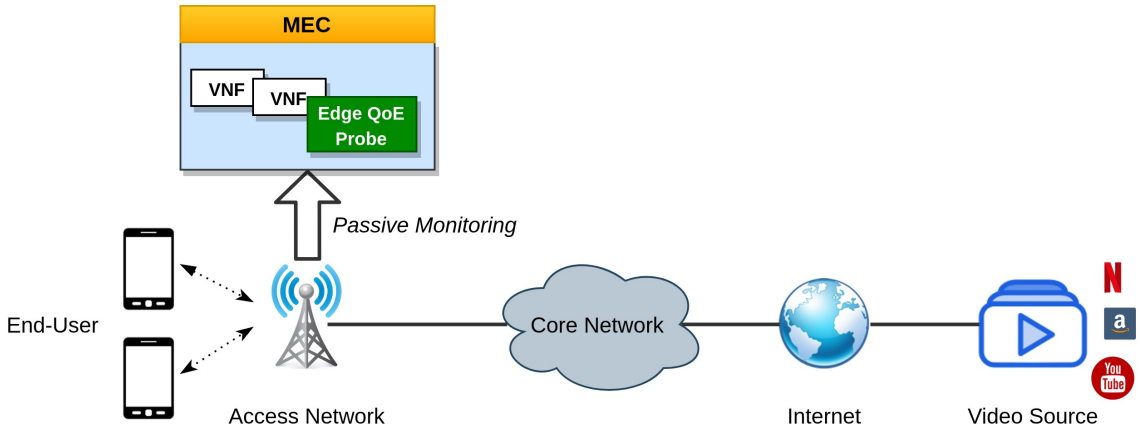


Figure 3.1: A High-level Network Architecture

Our proposed solution for the DASH video service tracks the bi-directional encrypted network traffic from the network edge, enable us to infer users' QoE according to the bi-directional network traffic's lightweight QoS features. Specifically, it continuously collects the bi-directional network traffic in a real-time fashion and predicts end-users QoE for DASH video service.

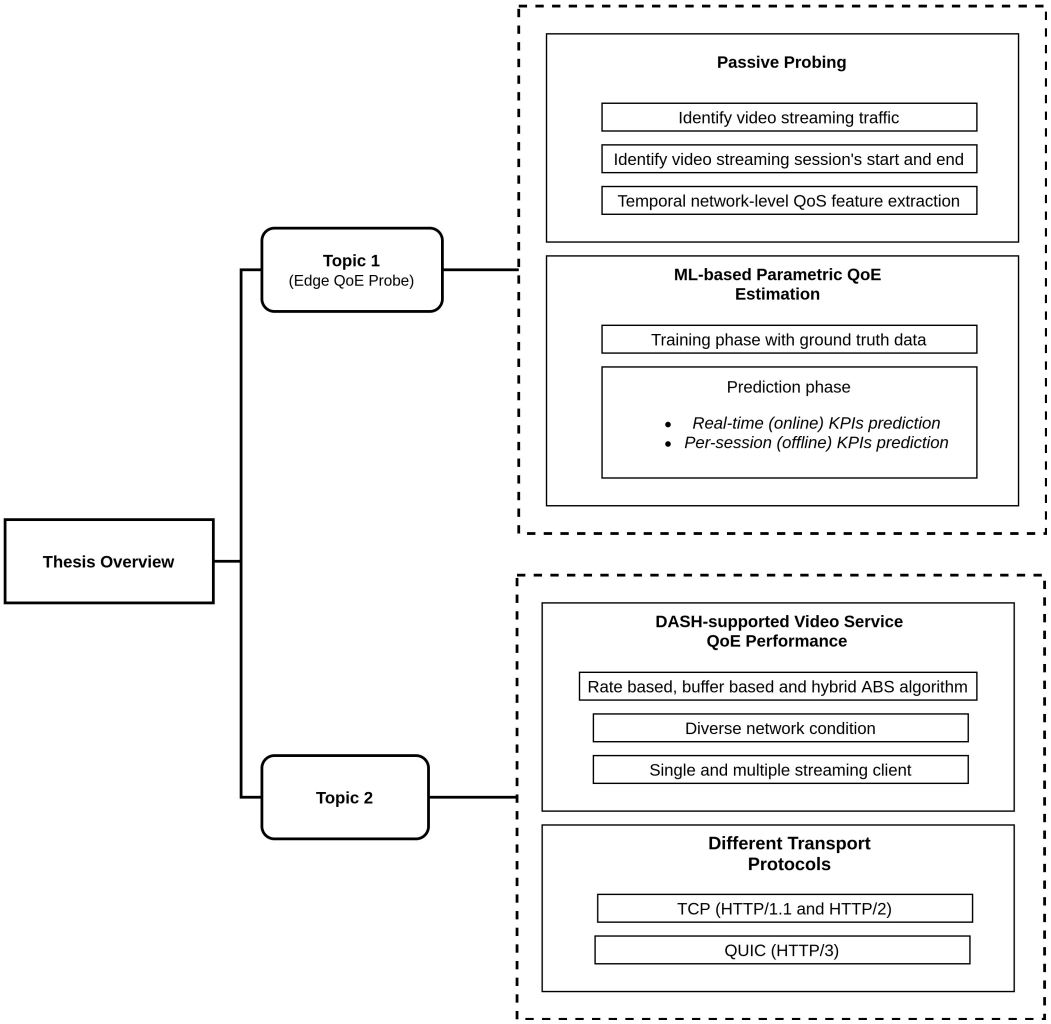


Figure 3.2: Overall Thesis Design Overview

However, this thesis is based on the prediction of the end-user QoE for the DASH video service at edge premises as well as the evaluation of the QoE performance of the DASH video service. Hence the overall design requirement is divided into two topics.

- **Topic 1**: QoE prediction of DASH-supported video service at edge premises.
- **Topic 2**: QoE performance evaluation of DASH-supported video service over different transport.

For two topics, we state **Topic 1** is mandatory for the `Edge QoE Probe` design and implementation, and **Topic 2** is non-mandatory or an additional part to analyze the QoE performance over the different transport protocols. Details on each topic are presented in Figure 3.2 and stated as follows.

3.2 Topic 1: QoE Prediction of DASH-supported Video Service at Edge Premises

This section discusses the steps to be considered in designing the `Edge QoE Probe` for passive network traffic detection and ML-based QoE estimation. Further, to implement the probing scheme (interchangeable with `Edge QoE Probe`), we thoroughly explain our controlled experiment.

3.2.1 Passive Probing-based Network Traffic Acquisition and Processing

Network traffic sent over the network system needs to capture without altering or modifying the original traffic. A packet sniffer can capture the network traffic at the access network/point interface (or any other network interface) on the network layer and processes the IP packets into QoS KPIs.

In general, network traffic comprises parallel traffic flows for several services and a network interface containing raw network packets. Moreover, application-level header information remains inaccessible due to the end-to-end encryption over TCP (HTTPs) and QUIC transport protocols. Hence, inferring video quality by the passive probing scheme from network traffic requires considering the following points to extract and process the QoS KPIs as features efficiently.

Identify video streaming service traffic. The first responsibility to identify corresponding video streaming service traffic from a network interface (e.g., edge node/access point). For this purpose, to capture the video service traffic, the probing scheme can utilize [55]-

- a unique 5-tuple (IP source, IP destination, port source, port destination, protocol).
- video content providers used IP address.
- inspect DNS responses (matching the DNS lookups against the known signature of video service).
- analyzes the Server Name Indication (SNI) in TLS handshakes.
- traffic classification techniques by ML.

Identify the start and end of the video streaming sessions. After identifying video streaming service traffic, the next requirement is to determine the video streaming sessions. The probing scheme should have characteristics that can pinpoint the video streaming session's start and end to obtain the quality or QoE of the video service at the right moment. But, in general, from the noisy network traffic, it is not a trivial task. To solve this, work in [57] showcase a heuristic solution where-

- a spike of non-video traffic such as player code or web-page catalog in the download link indicates the session's start.
- no video traffic or a silent period indicates the end of the session.

This technique is more viable when a DASH player streams video content one by one. The silent period in download link and later a spike in download link makes it convenient to recognize the video streaming sessions. Since each of the DASH players maintains a certain buffer threshold to store the downloaded video segments. As a result, it creates a silent period for download video segments after receiving all the video segments in the buffer.

Such a technique can trigger the probing scheme to identify the video session's start and end. Therefore, at the right moment, our approach can track the corresponding video session's network traffic and extract the network-level QoS features.

Temporal network-level QoS feature extraction. After identifying both video streaming traffic and video session, the subsequent role is to extract QoS features from the network traffic. Due to end-to-end traffic encryption, application-level information remains hidden in network traffic and makes it infeasible to inspect the corresponding video streaming's application-level information directly. In this situation, the probing scheme only can observe the bi-directional network traffic as opposed to time-sequence. Therefore, it is needed to extract the QoS feature over the time sequence.

To make the QoS extraction process time and memory efficient, the probing approach requires a lightweight method similar to work [59] [58] [60], which follows a temporal network-level features extraction solution in real-time. In this process, at each time interval (T), the network layer's multiple statistical features are extracted from the IP header information with different temporal aggregations of the *current* time window (T) and past time windows. Past time windows are the most recent N time window or *trend* window and past all-time window or *session* window. Both include the *current* window as well. More specifically, in the temporal feature extraction approach, the probing scheme can extract three types of time windows, i.e., *current*, *trend*, and *session* QoS features at each time interval.

We can choose the shortest possible size for the time window (e.g., 1-second, 5-second) for real-time QoE estimation. Furthermore, this approach only relies on the bi-directional (uplink and downlink) network-level information; thus, it does not require application-level segment-detection mechanisms and segment-based features, unlike work [57] [56]. Segment-detection and extract segment-based features require more processing time and impracticable if multiple segment downloads in parallel. Hence, without segment-detection, the QoS feature's temporal extraction is considered more robust, lightweight, and fine-grain.

3.2.2 ML-based Parametric QoE Estimation

The extracted network-level QoS features need to be fed into the supervised ML model to make the QoS-to-QoE correlation model and estimate end-user QoE. Therefore, the probe scheme needs ground truth from video streaming performance for labeling the dataset, which is a prerequisite to train a supervised ML model. Wherein the extracted QoS KPIs from the network edge premises will be used as a feature input. Hereby, our proposed `Edge QoE Probe` has two operational phases: training (a.k.a learning) and prediction (a.k.a inference).

1. **Training Phase**: In this phase, the probing scheme relies on a specific user who will contribute using a particular device or player by continuously reporting the video streaming performance in objective QoE KPIs format. Alternatively, an HTTP proxy will decrypt the encrypted traffic and read video quality information from the IP packet header. Later, this KPIs information converts into target QoE labels. Then, the probing scheme will train a supervised ML algorithm as a QoS-to-QoE correlation model based on converted QoE labels and extracted QoS features from the network edge.

Note that the training phase takes end-user devices or DPI assistance (which might seem to conflict with this thesis's motivation) because the supervised ML technique demands ground truth.

2. **Prediction Phase**: Afterward, in run time, the probe scheme will only use the trained model to predict or infer a group of users or a specific user QoE by extracting QoS features from the network edge as an input of the model. This phase no longer needs a user-side report or HTTP proxy to collect ground truth information.

Our proposed `Edge QoE Probe` uses the ML model for both real-time and per-session QoE estimation purposes.

- For real-time QoE estimation, a particular time window's network traffic temporal QoS features are used to predict that time window's QoE.
- On the contrary, for per-session QoE estimation, the entire video session network traffic (e.g., by aggregating real-time features) is used to predict the QoE of the whole video session.

Both real-time and per-session QoE estimation output has two different roles for QoE management in optimizing network performance, as shown in Figure 3.3.

- Real-time QoE estimation helps service assurance CCL for run-time network optimization with reactive performance diagnosis and resource allocation.
- Per-session QoE estimation helps network operators to review the SLA for proactive network capacity planning and configuration.

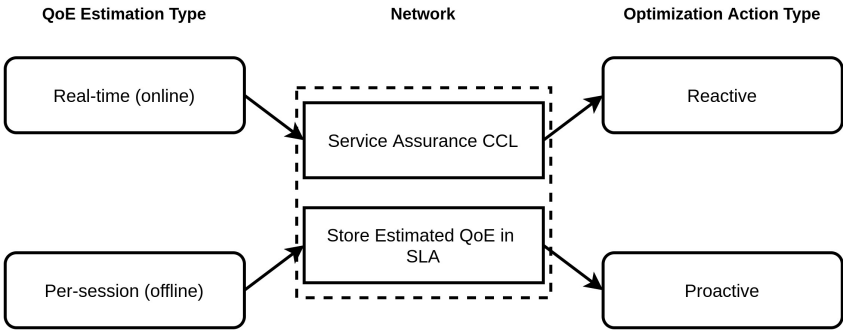


Figure 3.3: Twofold QoE Estimation Benefits

3.2.3 Implementation

In order to implement and evaluate the performance of the predictive `Edge QoE Probe`, we underwent a controlled experiment. The overall controlled experiment was based on four-section: *(i)* Experimental Setup *(ii)* Network-level Temporal QoS Features *(iii)* Application-level QoE KPIs as Ground Truth, and *(iv)* Supervised ML-based QoE Estimation. Each of the sections is explained below.

3.2.3.1 Experimental Setup

To generate and capture network traffic at the edge of the network and video streaming log at the client-side, we set up a testbed for the controlled experiment, which enclosed: *(i)* **Mininet-WiFi** - a network emulation tool, *(ii)* **goDASH** - a DASH video player, *(iii)* **Caddy** - a web server hosting DASH video content, *(iv)* **DITG** - a background traffic generator, *(v)* **Linux TC** - a traffic controller in the Linux kernel, and *(vi)* **Tcpdump** - a passive network traffic sniffer.

- **Network emulation.** Network emulation is a technique to mimic real network behavior over the virtual network. Since wireless networks are growing in popularity and SDN approaches to virtualized and programmable networks; thus, Mininet-WiFi [84] (a fork of Mininet[1] extended), a wireless SDN emulator was chosen for our work. It supports WiFi by adding virtualized WiFi Stations (STAs) and Access Points (APs). Therefore, we emulated a network scenario with a high-fidelity and fully controllable Mininet-WiFi emulation environment. All the experiments carried out inside of the virtual machine created with a virtual box. We used a network architecture scenario as shown in Figure 3.4, where the topology comprises one Access Point (AP) and one Open vSwitch (OvS)[2]. The topology depicts one DASH client and one DITG cross-traffic sender are connected with AP. On the opposite side, a single web server hosts DASH video content and one DITG cross-traffic receiver are connected with OvS.

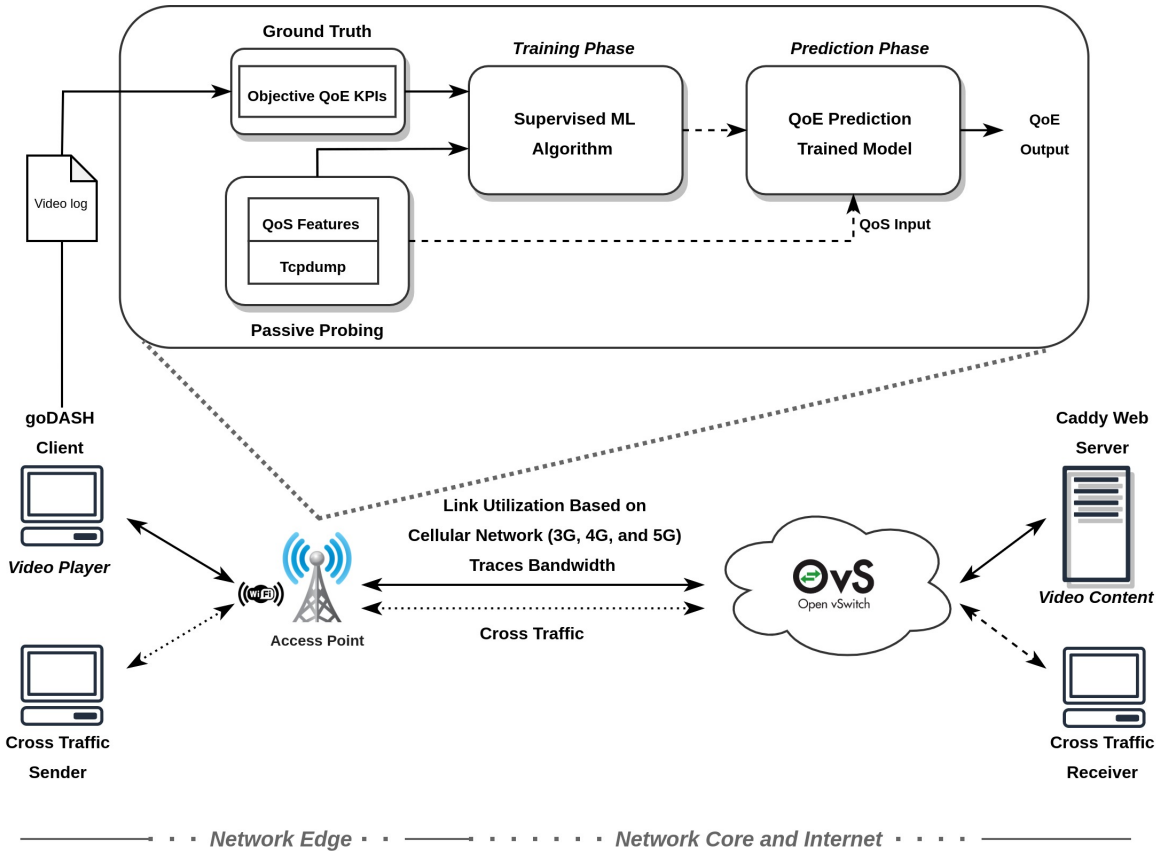


Figure 3.4: Network Emulation Architecture

[1]http://www.mininet.org
[2]https://www.openvswitch.org/

- **DASH player.** To stream a DASH-supported video, we used goDASH [85], a lightweight headless streaming player at the client-side. It streams video content without decoding, making goDASH lightweight and memory-efficient for large-scale evaluation than other players such as dash.js and ExoPlayer. goDASH has the feature of supporting numerous state-of-the-art ABS algorithms, two transport protocols (e.g., TCP and QUIC). Also, goDASH provides a video streaming log containing per segment objective QoE metrics (e.g., segment arrival time, stall, bit-rate, resolution) and five different real-time output from QoE models (ITU-T P.1203 MOS, Claey, Dunamu, Yin, and Yu). Since TCP (HTTPs) and QUIC require a secured and encrypted connection, goDASH was equipped with its goDASHbed[3] to set security certificates on both ends (client and server). Our testbed derives all the features of the goDASHbed. We customized the network topology, limiting the link bandwidth from network traces, DITG cross-traffic utilization, DASH video content, and ABS algorithm selection.

- **Web server and video source.** The testbed offered a Caddy[4] (v2) webserver hosting DASH video content. Caddy is a Web Server Gateway Interface (WSGI) server supporting HTTP/3 atop experimental QUIC and HTTP/1.1 and HTTP/2 atop TCP. Hence the DASH clients stream videos either over TCP or QUIC transport. Caddy leverage the quic-go[5] library, an open-source implementation of QUIC transport protocol (draft-29) [86] written in Go language. At the server-side, we used a 4-second segment duration short science fiction film (Tears of Steel), sourced from a publicly available 4K DASH video dataset [87]. This video content has a total duration of over 14 minutes, encoded using H.264/AVC, and contains eight resolutions across thirteen representation rates. The detailed mapping of the video resolution to bit-rate is shown in Table 3.1.

Table 3.1: Quality Representations: (Bit-rates vs Resolutions) *Tears of Steel* Video

Bit-rates (Kbps)	230	375	560	750	1050	1750	2350	3000	3850	4300	15000	25000	40000
Resolutions	320x 180	384x 216	512x 288	512x 288	640x 360	736x 414	1280x 720	1280x 720	1920x 1080	1920x 1080	1920x 1080	3840x 2160	3840x 2160

- **Trace-based link bandwidth utilization.** We emulated different network conditions through Linux TC (Hierarchical Token Bucket) between AP and OvS link using the downlink bandwidth parameter from 3G [88], 4G [89], and 5G [90] cellular network traces. For our experiment, we randomly selected a total of 15 different mobility traces where 3G (Bandwidth: mean=1.74 and std=0.92 Mbps), 4G (Bandwidth: mean=12.50 and std=14.64 Mbps), and 5G (Bandwidth: mean=44.41 and std=53.40 Mbps) traces represent low, moderate, and higher bandwidth scenarios respectively. Figure 3.5 illustrates the statistical overview of the 15 different traces.

- **Background traffic generator.** The DITG tool was used to introduce cross-traffic alongside the video streaming traffic in the testbed. Using DITG, we sent three concurrent flows of UDP traffic from one sender host to another receiver host. The amount of total UDP cross-traffic in such quantity occupied approximately 20% of the average bandwidth of 3G, 4G, and 5G traces.

[3]https://github.com/uccmisl/goDASHbed
[4]https://caddyserver.com
[5]https://github.com/lucas-clemente/quic-go

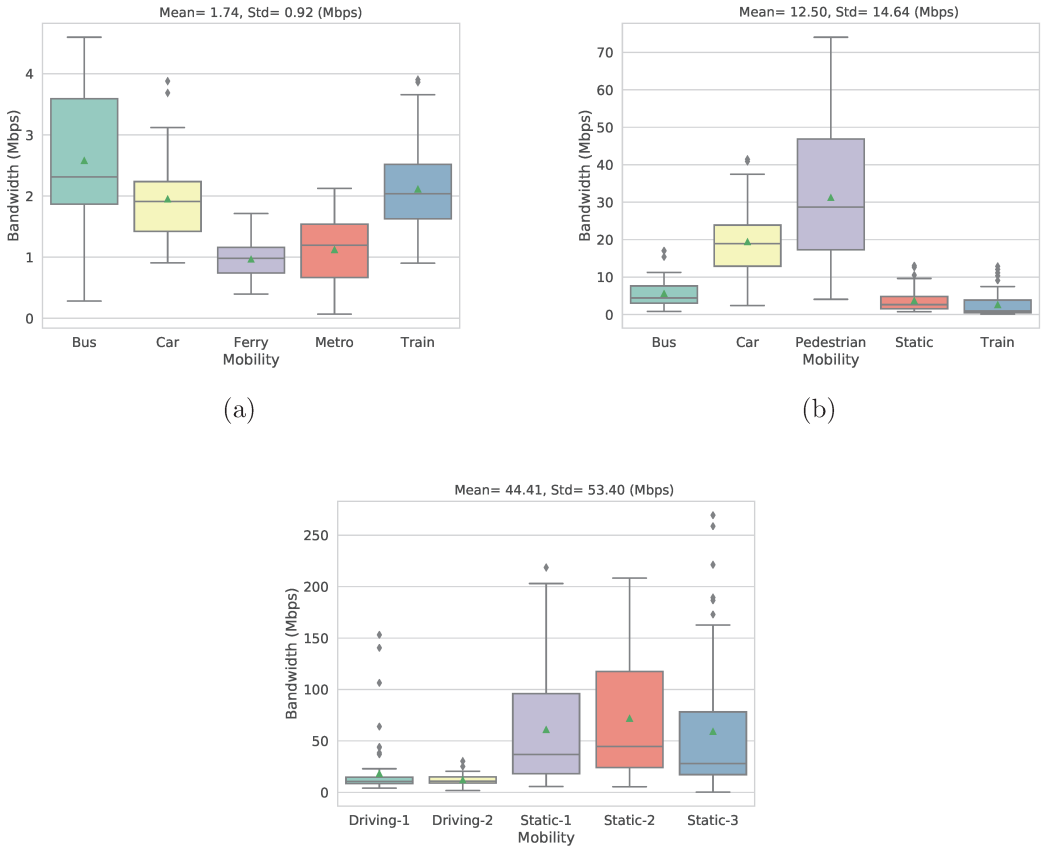


Figure 3.5: Fifteen Different Mobility Traces over (a) 3G, (b) 4G, and (c) 5G Networks Statistical Overview Through Box-plot

- **Passive network traffic monitoring at edge.** During the video streaming emulation to extract network-level QoS KPIs features, firstly, we captured the network traffic at the AP interface with Tcpdump[6], a packet sniffer tool. It passively grabs what is going on within the specific interface without intercept and alter the raw data. It is also flexible in selecting which traffic to capture (e.g., TCP, UDP) at which port and interface. Besides, it captures network traces as fast as possible in a lightweight manner as it requires less RAM usage than tshark/wireshark[7].
  - **Video streaming traffic identification.** The testbed contained two types of traffic- video streaming traffic and cross-traffic, as shown in Figure 3.6. We were already aware of both traffic's source and destination IP addresses due to the controlled experimental setup. Hence, at the AP interface, we quickly identified the IP address of the video streaming traffic used by the video server (Caddy) through Tcpdump. As our testbed provides end-to-end encryption for video streaming traffic, it makes the application header information inaccessible directly. The captured encrypted traffic traces (uplink and downlink) information by Tcpdump was stored as a dump file (PCAP). Later, only the network layer header information of offloaded PCAP files was analyzed to process network-level QoS KPIs features.

[6]https://www.tcpdump.org/manpages/tcpdump.1.html
[7]https://www.wireshark.org

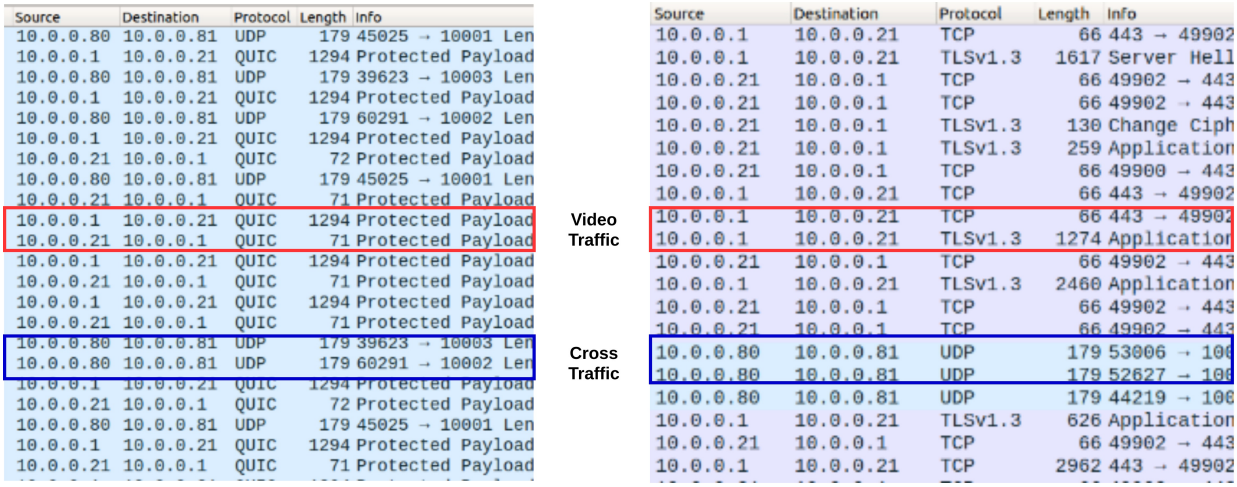


Figure 3.6: Testbed Generated Network Traffic

  - **Video streaming session starts and ends identification.** In our controlled experiment, we maintained a parameterized script to stream video one by one at each time. Thus, the testbed automatically started capturing network traffic when the video began to play and stopped capturing when the video streaming finished. For each video streaming experiment, the testbed captured and stored the PCAPs file separately.

- **Experimental parameter usage.** To stream video each time at client-side, goDASH player was configured with specific ABS algorithms. We used the following six state-of-the-art ABS algorithms from three categories in the entire experiment. The details of each category ABS algorithm are taken from work [91].

  - **Rate-based ABS algorithm**
    - **Conventional** [92] and **Exponential**: Both algorithms make the decision on the next segment by using an exponential moving average of past segments' delivery rate.
  - **Buffer-based ABS algorithm**
    - **BBA** [36] and **Logistic** [93]: Both algorithms select the next segment quality by mapping the buffer level to a target segment quality. Moreover, the Netflix platform's BBA scheme incorporates a throughput-based decision in its "startup" phase while considering future video segment sizes. Logistic uses a logistic model to map buffer levels to video bit-rate.
  - **Hybrid ABS algorithm**
    - **Arbiter** [94] and **Elastic** [95]: Arbiter uses the exponential weighted moving average of the last ten segment's delivery rates to estimate the next segment's available throughput. However, to adapt quickly to sudden changes in throughput, it uses adaptive scaling, such that its estimation is based on the second moment of the throughput sample. The additional scaling factor reflects the player's buffer state. On the other hand, Elastic uses a harmonic average of the recent five-segment rates. The harmonic average is a conservative estimate of available throughput. Furthermore, Elastic uses the control theory to combine throughput estimate and buffer levels when making video rate selection decisions.

During each streaming session, link bandwidth between AP and OvS was used based on individual traces from 15 selected cellular networks (3G, 4G, and 5G) traces, and the video content was played for up to 2 minutes. Moreover, TCP and QUIC transport were used separately for each streaming session. We customized the selected traces for every sample with 4-second granularity. Linux TC was applied to change the link bandwidth after every 4 seconds during the video streaming so that at least two segments can easily download between 4-second intervals. Each streaming session experiment with a specific ABS algorithm, link utilization, and transport option was repeated five times. All the parameters used for different combinations during video streaming sessions are given in Table 3.2.

Table 3.2: Experimental Parameters Usage Based on Different Combinations

Cellular Network	**Total Traces**	**Mobility**	**DASH Client**	**ABS Algorithms**	**Transport Option**	**Video Duration**	**Cross Traffic**	**Experiment Repetition**
3G	5	Bus Car Ferry Metro Train	1	Conventional Exponential BBA Logistic Arbiter Elastic	TCP QUIC	2 min	UDP	5 times
4G	5	Bus Car Pedestrian Static Train	1	Conventional Exponential BBA Logistic Arbiter Elastic	TCP QUIC	2 min	UDP	5 times
5G	5	Driving-1 Driving-2 Static-1 Static-2 Static-3	1	Conventional Exponential BBA Logistic Arbiter Elastic	TCP QUIC	2 min	UDP	5 times

3.2.3.2 Network-level Temporal QoS Features

A Python-based script was used to extract the network-level QoS KPIs features for QoE estimation. This script analyzed and calculated various network-level QoS KPIs statistics using only packet headers, specifically the IP addresses and packet sizes from traffic traces (PCAPs).

Real-time QoS features. For real-time, the core idea was to estimate QoE KPIs for each short interval time window, which should be constant during the entire video session. For this reason, temporal network-level features were calculated from bi-directional network traffic in a sliding window fashion. In this technique, for each short interval time window, the QoS KPIs features were calculated for that time window as well as related past windows. Each short interval was defined as a *current* time window, and past windows were defined as two ways-most recent windows or *trend* time window and past all windows or *session* window.

Throughout our work, we chose a half-second or 0.5-second interval as a *current* window length. For the *trend* window, the time window length was considered 1, 3, 5, 10, and 20 seconds. Note that the *trend* window contains traffic information on the *current* window as well. In contrast, the *session* window covers all the previous time window's traffic information from the beginning of the video stream to till *current* time window.

For each type of time window, network layers QoS KPIs features were calculated for uplink and downlink directions and TCP and QUIC connections. Specifically, four basic network-level QoS KPIs: **throughput**, **number of packets**, **packet sizes**, and **packets inter-arrival time** were considered throughout our work. Later, for each time window, we computed the overall statistics of each KPI. The detailed description of the calculated features is presented in Table 3.3. The following format was used to name each real-time QoS features-

traffic-direction__kpi__statistic__time-window

Thus, for real-time QoE KPIs estimation, we extracted a total of 252 features for the TCP connection and 168 features for the QUIC connection.

Entire session's QoS features. As opposed to real-time QoE estimation, the probe scheme estimated the entire video session's QoE as well. For this reason, calculated statistics of the QoS features i.e., total numbers of packet, average throughput, mean packet size and inter-arrival time in the *current* time window were aggregated for entire video session traffic. The following format was used to name each QoS features for entire session-

traffic-direction__kpi__statistic__S (entire session)

Thus, we calculated 216 features for the TCP connection and 144 features for the QUIC connection for the entire video session. Besides, the detailed description of the per video session QoS features is presented in Table 3.4. An overview of the real-time and per-session QoS features computing approach is given in Figure 3.7.

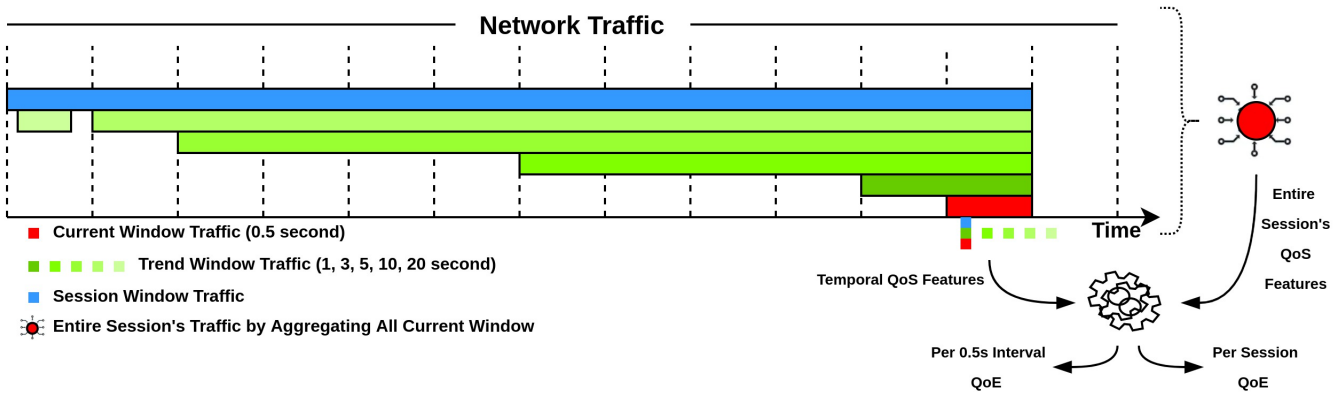


Figure 3.7: Network-level QoS Features Extraction Technique

3.2.3.3 Application-level QoE KPIs as Ground Truth

A prerequisite for supervises ML technique is to use QoE metrics/KPIs as ground truth. Therefore, it is necessary to obtain QoE metrics while a video is played from the client-side in the emulation-based experiment.

During our experiment, goDASH was used as a client-side player to stream the video content. It provides a log file of streaming content which consists of potential information about each downloaded segment such as bit-rate, buffer level, stall duration, delivery rate, actual rate. These information are regarded enough for QoE metrics (e.g., the average video quality, quality switching rate, and video stalls) calculation [96] [97]. Moreover, goDASH provides optional information (e.g., codec, frame rate, abs algorithm, segment duration) at the log file for each segment with five QoE models output- ITU-T Rec. P.1203 MOS Score [98] [99], Clay [100], Dunamu [101], Yin [102], and Yu [103].

These pieces of optional information are used as input of recently standardized ITU-T recommendation P.1203 QoE model [99] to obtain MOS (derived from subjective MOS studies)

Table 3.3: Real-time QoS Features Statistics Calculated for Uplink and Downlink Direction Traffic and on Current (0.5 second), Trend (1, 3, 5, 10, and 20 second), and Session Window

Features	Connection Type	Description
UP/DL_**Th_avg**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **average throughput** in the window
UP/DL_**Pkt-N_total**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **total number of packets** in the window
UP/DL_**Pkt-N-g100_total**_CW/TW/SW	Only TCP	Uplink and Downlink **total number of packets** in the window (packets greater than 100B are only counted to **ignore ACKs**)
UP/DL_**Pkt-S_mean**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **mean packet size** in the window
UP/DL_**Pkt-S_medn**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **median packet size** in the window
UP/DL_**Pkt-S_std**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **standard deviation of the packet size** in the window
UP/DL_**Pkt-S_max**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **maximum packet size** in the window
UP/DL_**Pkt-S_min**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **minimum packet size** in the window
UP/DL_**Pkt-S-g100_mean**_CW/TW/SW	Only TCP	Uplink and Downlink **mean packet size** in the window (ignore ACKs)
UP/DL_**Pkt-S-g100_medn**_CW/TW/SW	Only TCP	Uplink and Downlink **median packet size** in the window (ignore ACKs)
UP/DL_**Pkt-S-g100_std**_CW/TW/SW	Only TCP	Uplink and Downlink **standard deviation of the packet size** in the window (ignore ACKs)
UP/DL_**Pkt-S-g100_max**_CW/TW/SW	Only TCP	Uplink and Downlink **maximum packet size** in the window (ignore ACKs)
UP/DL_**Pkt-S-g100_min**_CW/TW/SW	Only TCP	Uplink and Downlink **minimum packet size** in the window (ignore ACKs)
UP/DL_**IA_mean**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **mean inter-arrival time of packets** in the window
UP/DL_**IA_medn**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **median inter-arrival time of packets** in the window
UP/DL_**IA_std**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **packet inter-arrival time's standard deviation** in the window
UP/DL_**IA_max**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **maximum inter-arrival time of packets** in the window
UP/DL_**IA_min**_CW/TW/SW	TCP, QUIC	Uplink and Downlink **minimum inter-arrival time of packets** in the window

Table 3.4: Entire Session's QoS Features Statistics Calculated for Uplink and Downlink Direction Traffic by Aggregating Current Window's QoS Features

Features	Connection Type	Description
UP/DL_**Th_avg/medn/std/max/min_S**	TCP, QUIC	Uplink and Downlink **average, median, standard deviation, maximum, minimum throughput** in the entire video session
UP/DL_**Th_avg-f25/f50/l25/l50_S**	TCP, QUIC	Uplink and Downlink **average throughput in the first and last 25% and 50%** of video session
UP/DL_**Th_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	TCP, QUIC	Uplink and Downlink **10 to 90 percentile throughput** in the entire video session
UP/DL_**Pkt-N_total/mean/medn/std/max/min_S**	TCP, QUIC	Uplink and Downlink **total, mean, median, standard deviation, maximum, minimum of packet numbers** in the entire video session
UP/DL_**Pkt-N_total-f25/f50/l25/l50_S**	TCP, QUIC	Uplink and Downlink **total number of packets in the first and last 25% and 50%** of video session
UP/DL_**Pkt-N_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	TCP, QUIC	Uplink and Downlink **10 to 90 percentile of total packets** in the entire video session
UP/DL_**Pkt-N-g100_total/mean/medn/std/max/min_S**	Only TCP	Uplink and Downlink **total, mean, median, standard deviation, maximum, minimum of packet numbers** in the entire video session (ignore ACKs)
UP/DL_**Pkt-N-g100_total/f25/f50/l25/l50_S**	Only TCP	Uplink and Downlink **total number of packets in the first and last 25% and 50%** of video session (ignore ACKs)
UP/DL_**Pkt-N-g100_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	Only TCP	Uplink and Downlink **10 to 90 percentile of total packets** in the entire video session (ignore ACKs)
UP/DL_**Pkt-S_mean/medn/std/max/min_S**	TCP, QUIC	Uplink and Downlink **mean, median, standard deviation, maximum, minimum packet size** in the entire video session
UP/DL_**Pkt-S_mean-f25/f50/l25/l50_S**	TCP, QUIC	Uplink and Downlink **mean packet size in the first and last 25% and 50%** of video session
UP/DL_**Pkt-S_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	TCP, QUIC	Uplink and Downlink **10 to 90 percentile of packet size** in the entire video session
UP/DL_**Pkt-S-g100_mean/medn/std/max/min_S**	Only TCP	Uplink and Downlink **mean, median, standard deviation, maximum, minimum packet size** in the entire video session (ignore ACKs)
UP/DL_**Pkt-S-g100_mean-f25/f50/l25/l50**	Only TCP	Uplink and Downlink **mean packet size in the first and last 25% and 50%** of video session (ignore ACKs)
UP/DL_**Pkt-S-g100_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	Only TCP	Uplink and Downlink **10 to 90 percentile of packet size** in the entire video session (ignore ACKs)
UP/DL_**IA_mean/medn/std/max/min_S**	TCP, QUIC	Uplink and Downlink **mean, median, standard deviation, maximum, minimum of packet inter-arrival time** in the entire video session
UP/DL_**IA_mean-f25/f50/l25/l50_S**	TCP, QUIC	Uplink and Downlink **mean packet inter-arrival time in the first and last 25% and 50%** of video session
UP/DL_**IA_p-10/p-20/p-30/p-40 /p-50/p-60/p-70/p-80/p-90_S**	TCP, QUIC	Uplink and Downlink **10 to 90 percentile of packet inter-arrival time** in the entire video session

for each segment. The recommendation describes four different quality modules as mode-0, 1, 2, and 3. Each mode of operation has different input requirements ranging from the only log files to segment level information of the video stream. goDASH uses the mode-0 ITU-T P.1203 QoE model that requires the bit-rate, codec, duration, frame rate, and resolution of each segment to obtain the MOS. The main shortcoming of the ITU-T standardization is its limitation to H.264 (AVC) encoding content to Full HD (1920x1080). Hence in our experiment, each video streaming was conducted by setting the maximum 1920x1080 resolution in the goDASH player end.

Table 3.5 presents the options for per segment output logs generated by the goDASH. The logs are divided into three parts. Part 1- default output, Part 2- optional output, and Part 3- QoE models output. Table 3.6 (default output) and 3.7 (optional and QoE models output) present the first five-segments information obtained from the goDASH log file for arbiter algorithm, 4-second segment duration video content, and QUIC (HTTP/3) protocol.

Table 3.5: Notation Used in the goDASH Trace Output Logs

Part	Type	Description
1. Default output	Seg_#	Streamed segment number
	Arr_Time	Arrival time in milliseconds (ms)
	Del_Time	Time taken to receive the segment (ms)
	Stall_Dur	Stall duration (ms)
	Rep_Level	Representation quality of downloaded segment (kbps) (taken from MPD file)
	Del_Rate	Delivery rate of downloaded segment (kbps) (segment size divided by time for delivery)
	Act_Rate	Actual rate of of downloaded segment (kbps)(segment size divided by the segment duration)
	Byte_Size	Segment size in bytes
	Buff_Level	Buffer level (ms)
2. Optional output	Algorithm	ABS algorithm name
	Seg_Dur	Segment duration (ms)
	Codec	Video encoder
	Width	Representation width in pixels
	Height	Representation height in pixels
	FPS	Frame rate of the streamed video
	Play_Pos	Current playback position (ms)
	RTT	Application level (ms)
	Protocol	HTTP protocol
3. Five QoE model output	P.1203	P.1203 standard - scale [0, 5]
	Clae	Clae model - scale [0, 5
	Duanmu	Duanmu model - scale [0, 100]
	Yin	Yin model - scale dependent on HAS bitrates
	Yu	Yu model - scale [0, 5]

Table 3.6: Sample Default Trace Output from goDASH for First 5 Segments - Arbiter Algorithm, 4-second Segment Duration and QUIC (HTTP/3) Protocol

Seg_#	Arr_time	Del_Time	Stall_Dur	Rep_Level	Del_Rate	Act_Rate	Byte_Size	Buff_Level
1	79	78	0	236	565	11	5516	4000
2	717	153	0	375	586	22	11219	8000
3	6192	5005	0	375	570	714	357007	6525
4	10718	4037	0	236	493	498	249220	5999
5	12023	772	0	236	1284	247	123943	8694

Real-time QoE estimation ground truth. Note that goDASH provides only the required output per video downloaded segment. We need video streaming performance information (QoE KPIs) with a time sequence for real-time QoE prediction. For this reason, we customized

Table 3.7: Sample Optional and QoE Models Output from goDASH for First 5 Segments - Arbiter Algorithm, 4-second Segment Duration and QUIC (HTTP/3) Protocol

Seg_#	Algorithm	Seg_Dur	Codec	Width	Height	FPS	Play_Pos	RTT	Protocol	P.1203	Claey	Duanmu	Yin	Yu
1	arbiter	4000	h264	320	180	24	0	28.679	HTTP/3	1.871	0.000	46.453	-11763.802	0.236
2	arbiter	4000	h264	384	216	24	4000	32.051	HTTP/3	1.871	0.420	37.839	-23527.604	0.306
3	arbiter	4000	h264	384	216	24	8000	46.048	HTTP/3	1.888	0.479	38.002	848.121	0.329
4	arbiter	4000	h264	320	180	24	12000	38.941	HTTP/3	1.876	0.449	37.839	944.792	0.306
5	arbiter	4000	h264	320	180	24	16000	34.004	HTTP/3	1.874	0.437	37.742	1180.990	0.292

the goDASH provided log information by subdividing the video streaming into a sequence of time. Throughout the work, we considered 0.5-second constant time length's QoE KPIs (e.g., playback status, stall event, bit-rate, and resolution) information based on per segment arrival information at the goDASH player end. goDASH, by default, used two segments for the initial buffer threshold to start playback; thus, by measuring the first two segments' arrival time, we interpreted whether the video started to play or not.

Table 3.8 presents customized goDASH output (8-second) for per 0.5-second interval video streaming. If we recall Table 3.6 we can see the first two segments taking 717 milliseconds or 0.717 seconds to arrive at the player buffer. Therefore, we can state that the 1st segment of the video was started to play at 0.717 seconds. The following step (See Algorithm 1) is taken to shape the time scale to ease the converting process[8] of per-segment goDASH output into a time sequence format. The last three-row in Table 3.8 showcase the scenario if any segment suffers stall.

Algorithm 1: The Procedure of Shaping the Time Scale

```
/* X = 0, 1, 2,...N and Y = 1, 2, 3,....N+1                    */
if time_range(s) ←[X.0, X.5] then
    if time(s) ≤ X.25 then
        time(s) ← X.0
    else
        time(s) ← X.5
else if time_range(s) ←[X.5, Y.0] then
    if time(s) ≤ X.75 then
        time(s) ← X.5
    else
        time(s) ← Y.0
```

Per-session QoE estimation ground truth. During each video streaming emulation time, we streamed video content for up to 2 minutes. Therefore, for the entire video session's ground truth, we measured the average values of 30 segments (4 seconds * 30 = 120 seconds or 2 minutes) goDASH provided outputs for a few QoE KPIs such as average resolution, average bit-rate (Rep_Level), average MOS. We also calculated other QoE KPIs from the summary of 30 segments output, such as stall ratio, quality switches.

[8](a) To calculate the segment playing start time, (b) If any segment got a stall, shape the stall time duration following the same step and then add with segment playing start time

Table 3.8: Sample QoE KPIs Output from Customized goDASH for First 8 Seconds (Except Last Three Row) - Arbiter Algorithm, 4-second Segment Duration and QUIC (HTTP/3) Protocol.

Time (s)	Playback Status	Segment	Playing Position (s)	Actual Segment Playing Start Time (s)	Customize Segment Playing Start Time (s)	Stall Event	Stall Duration Actual Time (s)	Stall Duration Customize Time (s)	Bit-rate (kbps)	Resolution
0.0	Not started	-	-	-	-	-	-	-	-	-
0.5	Played	1	0.0 to 0.4	**0.717**	**0.5**	No	**0.0**	**0.0**	236	320x180
1.0	Played	1	0.5 to 0.9	-	-	No	-	-	236	320x180
1.5	Played	1	1.0 to 1.4	-	-	No	-	-	236	320x180
2	Played	1	1.4 to 1.9	-	-	No	-	-	236	320x180
2.5	Played	1	2.0 to 2.4	-	-	No	-	-	236	320x180
3.0	Played	1	2.4 to 2.9	-	-	No	-	-	236	320x180
3.5	Played	1	3.0 to 3.4	-	-	No	-	-	236	320x180
4.0	Played	1	3.5 to 3.9	-	-	No	-	-	236	320x180
4.5	Played	2	4.0 to 4.4	**4.717**	**4.5**	No	**0.0**	**0.0**	375	384x216
5.0	Played	2	4.5 to 4.9	-	-	No	-	-	375	384x216
5.5	Played	2	5.0 to 5.4	-	-	No	-	-	375	384x216
6.0	Played	2	5.4 to 5.9	-	-	No	-	-	375	384x216
6.5	Played	2	6.0 to 6.4	-	-	No	-	-	375	384x216
7.0	Played	2	6.4 to 6.9	-	-	No	-	-	375	384x216
7.5	Played	2	7.0 to 7.4	-	-	No	-	-	375	384x216
8.0	Played	2	7.5 to 7.9	-	-	No	-	-	375	384x216
8.5	Paused	2	7.9	-	-	Yes	**0.82**	**1.0**	375	384x216
9.0	Paused	2	7.9	-	-	Yes	-	-	375	384x216
9.5	Played	3	8.0 to 8.4	**9.537**	**9.5**	No	-	-	750	512x288

3.2.3.4 Supervised ML-based QoE Estimation

QoE metrics/KPIs. To build supervised ML models that predict video performance both for real-time and per-session, we considered the following key QoE metrics/KPIs, which are commonly used to assess user-perceived video quality.

- **Resolution**: Video resolution determines the amount of detail in the displayed video streaming on the device. Resolution measures by the number of distinct pixels displayed in each dimension. The dimensions are usually quoted as "width `X` height". For ML-based resolution estimation, we can compute the average resolution of a video session [33] or a time window [55].

- **Bit-rate**: Bit-rate over a video session is a crucial metric for a displayed video streaming quality (e.g., resolution). Precisely, bit-rate is measured based on the amount of downloaded segment data during the video session. The ABS algorithms aim to maximize video quality by adopting a high bit-rate so that end-user perceived better quality of the video. However, bit-rate has a complex relation with quality (e.g., resolution) as bit-rate depends on the encoding and content type [104]. On the other hand, only the bit-rate can not depict the actual user satisfaction [105]; other metrics such as startup delay, stall, and the number of quality (e.g., resolution) changes require to assess the accurate QoE.

- **Quality Switches**: This metric defines the number of quality (e.g., resolution) levels switched over video streaming. In general, poor network condition leads to a high number of changes in quality level. Such amplitude and frequency of quality switching can negatively influence end-user experience and affect QoE [106]. Two concurrent video streaming sessions might have the same bit-rate, but the session with the lower quality switch will be perceived better by the viewer.

- **Stall Ratio**: Stall occurs when there is no adequate number of video segments buffered, and the player stops the playback. Such a stall event harms QoE, and the higher number of stalling events can increase the user's abandonment rate when streaming the video [25]. In our experiment, the metric stall ratio is calculated as follows [33]:

$$Stall\ Ratio = \frac{Sum\quad of\quad Total\quad Stall\quad Duration}{Entire\quad Video\quad Duration} * 100$$

  The "stall duration" means that streaming pauses during the playback to fill up the buffer again.

- **Startup Delay**: The startup delay of video plays a vital role in the user's abandonment rate when playing the video [107]. The delay measured in the startup phase of video streaming is known as startup delay. Each of the DASH players has an initial buffer threshold. Before starting playback DASH video, the client downloads multiple video segments. When a certain number of downloaded segments hits that buffer threshold, the player initiates to start the video. In our experiment, we set 2 segments as an initial buffer threshold. That means the player starts playback when initially two-segment download. We measured the first two segments' arrival time in the player buffer for the startup delay metric.

- **P1203 MOS**: Apart from the above-stated metrics, we also considered the subjective MOS values ranging between 1 and 5, according to ITU-T Rec. P.1203.1 standardization

(an objective parametric model derived from subjective studies). The goDASH player used in our experiment has the feature to calculate per-segment MOS based on the ITU-T Rec. P.1203.1 (mode-0) standardization[9] considering corresponding segments bit-rate, framerate, and resolution.

Supervised ML algorithms. Then, to evaluate different ML-model performance throughout the work, we used the following supervised ML algorithms[10] for classification and regression task, which are:

- **Linear Regression (Linear Model)**: To estimate continuous outcomes in the regression analysis, Linear Regression (Ordinary Least Squares) fits a linear model with coefficients to minimize the sum of squared residuals between the observed targets in the dataset and the targets predicted by the linear approximation.
- **Logistic Regression (Linear Model)**: Logistic Regression technique fits a linear model to perform the classification task or compute discrete outcomes. This technique analyzes a dataset in which one or more independent variables determine a discrete outcome using a logistic (or sigmoid) function[11]. There are two types of classification in supervised ML: Binary Classification (two discrete outcomes) and Multi-class Classification (more than two discrete outcomes).
- **K-Nearest Neighbors (Nearest Neighbors)**: K-Nearest Neighbors (KNN) supervised ML technique is used for both classification (KNC) and regression (KNR) tasks. The Nearest Neighbors method's logic is to find a predefined number of training samples closest in the distance to the new point and predict the outcome from these. In KNN, the number of samples is a constant value specified by the user.
- **Gaussian Naive Bayes (Naive Bayes)**: Naive Bayes is a classification-based technique with strong independence assumptions between the features. Specifically, the value of a particular feature is independent of the value of any other feature. Gaussian Naive Bayes (GNB) is a particular type of Naive Bayes. It supports continuous-valued features and assumes all the features follow Gaussian (normal) distribution.
- **Support Vector Machine**: Support Vector Machine (SVM) is the popular supervised ML method used for both classification (SVC) and regression (SVR), but mainly used for classification purposes. In the classification task, SVC works by finding a hyperplane in n-dimensional space (where n is the number of features) that distinctly classifies the data points. In contrast, SVR uses the same principles for the regression task, with only a few minor differences.
- **Decision Tree**: Decision Tree (DT) is a non-parametric supervised ML method used for both classification (DTC) and regression (DTR). It works by making a model that predicts a target variable's output by applying simple decision rules (if-else/yes-no conditions) inferred from the dataset's features.
- **Random Forest (Ensemble)**: Random Forest (RF) is also a supervised ML technique used for a diversity of tasks, including regression (RFR) and classification (RFC). RF

[9]https://github.com/itu-p1203/itu-p1203
[10]https://scikit-learn.org/stable/supervised_learning.html
[11]https://en.wikipedia.org/wiki/Logistic_function

technique follows an ensemble method; precisely, an ensemble of decision trees creates the RF model. In this method, an RF model is consists of a large number of small decision trees, known as estimators, where each produces its predictions. The RF model combines the predictions of the estimators to build a more accurate forecast.

- **Multi-layer Perceptron (Neural Network)**: Multi-layer Perceptron (MLP) is one of the extensively used supervised ML architectures in the Artificial Neural Network (ANN) area for classification (MPC) and regression (MPR). MLP uses a backpropagation[12] algorithm to fit the model, and it usually consists of three layers: an input layer, a hidden layer (s), and an output layer.

Model performance metrics. Classification- this approach is used to predict the dependent discrete/categorical output variable from different independent variables (input features). To evaluate the classification-based trained model, we used multiple matrices, including accuracy, precision, recall, F1-score. To present the predicted and actual class labels from the ML models, confusion matrix, an intuitive method is used as follows (Figure 3.8), which will help define each performance metric.

Actual / *Predicted*	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Figure 3.8: Confusion Matrix

Where, **T**rue **P**ositive (actual-1, predicted-1), **F**alse **N**egative (actual-1, predicted-0), **F**alse **P**ositive) (actual-0, predicted-1), and **T**rue **N**egative (actual-0, predicted-0)

- **Accuracy**: It is defined as the ratio of correctly predicted observation to the total observations and obtained using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Precision**: It determines the ratio of correctly predicted positive observations to the total predicted positive observations using the following formula:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: It determines the ratio of correctly predicted positive observations to all observations in the actual class that should have been identified as positive using the following formula:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-score**: It used to define the harmonic mean of precision and recall using the following formula:

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

[12]https://en.wikipedia.org/wiki/Backpropagation

Regression- the metrics used to evaluate the regression-based trained model are different from classification metrics and can predict the dependent variable's continuous values by learning from numerous independent features. In this work, we utilized the two most widely used metrics for the regression-based model. They are R^2 and RMSE.

- **R^2**: It is used to determine how well the trained model fits the regression line of dependent variables. It is also known as the Coefficient of Determination. Specifically, R^2 represents the variability of the dependent variable fit data compared with the mean of data and a horizontal line at the mean. Mathematically it defines as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

  where, y_i represents actual output value, $\hat{y}_i$ represents predicted output value by regression model, and $\bar{y}_i$ represents mean of actual output value. The closer the value of R^2 to 1 (or 100%), the better the model that accurately fits the actual and predicted value. On the other hand, the negative value of R^2 represents a model that fits the regression line worse than a horizontal line.

- **RMSE**: Root Mean Square Error or RMSE is the square root of the averaged squared difference (error) between the predicted value and actual value. Expressed as follows:

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

  where, y_i represents actual output value, and $\hat{y}_i$ represents predicted output value by regression model.

3.3 Topic 2: QoE Performance Evaluation of DASH-supported Video Service over Different Transport

This section aims to analyze the impact of TCP and QUIC transport on QoE, specifically for DASH video service. To assess DASH-based empirical QoE study over TCP and QUIC, we make a head-to-head QoE performance comparison of ABS algorithms over newly standardized QUIC and traditional TCP transport protocol in a controlled testbed environment. Throughout the evaluation, we try to find out a few answers to the following questions.

- **State-of-the-art ABS Algorithm's Behavior**: How does the different state-of-art ABS algorithm act over QUIC and traditional TCP?
- **Different Clients' Streaming Reaction**: Does QUIC outperform TCP when multiple clients are competing for the video content?
- **Stable and Unstable Network Scenario's Impact**: Does QUIC outperform TCP on diverse network conditions?

We leverage the similar approaches conducted in earlier work [76] [75] with our testbed incorporating the 5G network in addition to the 3G and 4G network and DASH content from Ultra High Definition (UHD) 4K dataset. For this purpose, we carry the following experiments.

1. QoE performance of ABS algorithms over QUIC and TCP across selected 3G, 4G, and 5G network traces (stable) for the single and parallel client that competes for DASH content.

2. QoE performance of ABS algorithms over QUIC and TCP on a restless network (unstable) with extreme bandwidth fluctuation.

To implement the two experiments mentioned above, we made a controller testbed for experimental purposes. The description of our experimental setup is given in the following subsection.

3.3.1 Experimental Setup

Our testbed framework was based on the following components: *(i)* **Mininet** - a network emulation tool, *(ii)* **goDASH** - a DASH video player, *(iii)* **Caddy** - a web server hosting DASH video content, *(iv)* **DITG** - a background traffic generator, and *(v)* **Linux TC** - a traffic controller in the Linux kernel.

- **Network emulation.** The Mininet network emulation tool was used to create the topology consist of two OvS (Switch 1 and 2). Figure 3.9 depicts DASH clients and one DITG cross-traffic sender are connected with Switch 1, and on the opposite side, the DASH web server and one DITG cross-traffic receiver are connected with Switch 2.

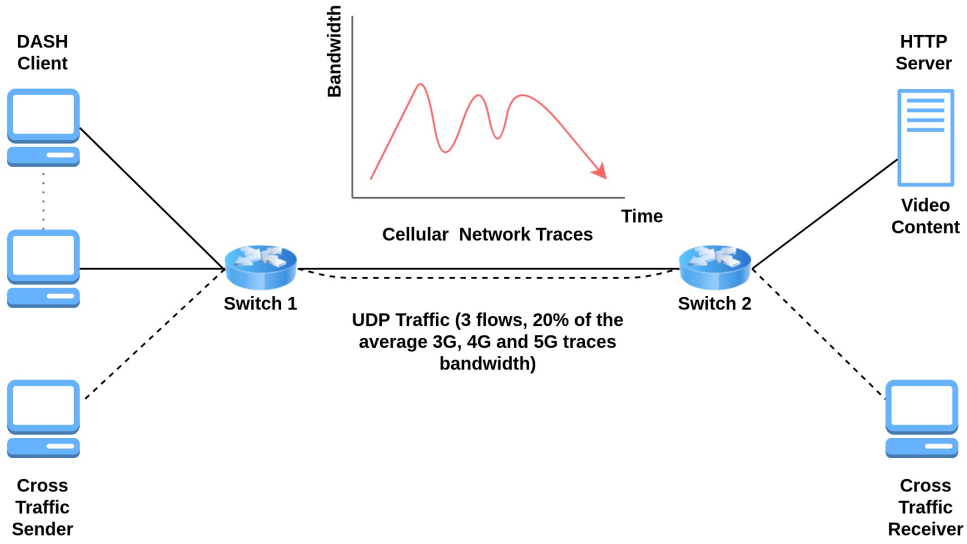


Figure 3.9: Network Emulation Topology

- **DASH player, web server, and video source.** Likewise Topic 1, we used goDASH, a lightweight headless player, to stream video from the client-side. On server-side. Caddy was used to offering an HTTP server with support HTTP/3 atop experimental QUIC and HTTP/1.1 and HTTP/2 atop TCP. Moreover, the server offered the same video content (Tears of Steel) used in the Topic 1 experiment.

- **Trace-based link bandwidth utilization, and background traffic generator.** Numerous network traces downlink bandwidth parameter was used to change the bandwidth condition between Switch 1 and 2 links through Linux TC (Hierarchical Token Bucket). For experiment (1), we used 15 different mobility pattern traces considering stable networks (used in Topic 1 experiment) as illustrated in Figure 3.5. Apart from this, for

experiment (2), another specific driving mobility pattern of the 5G network trace was used to emulate the behavior of an unstable network, where network mode frequently changes from 5G to LTE or HSPA+, as presented in Figure 3.10. However, using the DITG tool, we sent three concurrent flows of UDP traffic from one sender host to another receiver host to produce the cross-traffic. The amount of total UDP cross-traffic in such quantity occupied approximately 20% of the average network traces bandwidth (for experiment 1- five different traces' average bandwidth for each 3G, 4G, and 5G network; for experiment 2- one specific trace's average bandwidth for 5G network).

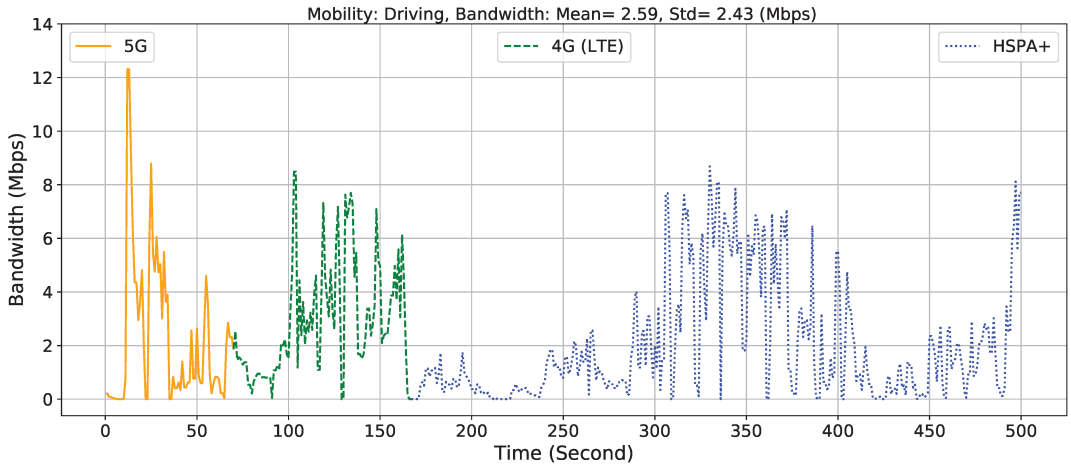


Figure 3.10: Single Selected Driving Mobility Trace over 5G Network Contains Extreme Bandwidth Fluctuation and Frequent Connection Changes

- **Experimental parameter usage.** For the sake of our QoE evaluation, we worked with the following three algorithms: *(i)* **Conventional (Rate-based) [92]**, *(ii)* **BBA (Buffer-based) [36]**, and *(iii)* **Arbiter (Hybrid) [94]**. As the current goDASH version does not support HTTP/2 standard, therefore for both experiments, goDASH clients played the video using the above mentioned three ABS algorithms from the DASH server (Caddy) using HTTP/1.1 atop TCP and HTTP/3 atop QUIC transport. During each streaming session, the video content was played for up to 3 minutes. Both experiments were repeated three times for unbiased statistical evaluation, and the QoE metrics' only average values were taken. All the parameters used for different combinations during video streaming sessions are given in Table 3.9.

- **QoE metrics/KPIs.** We considered the following five metrics that play an essential role in defining video streaming performance and end user's QoE: *(i)* **Average Bit-rate**, *(ii)* **Quality Switches**, *(iii)* **Stall Ratio**, *(iv)* **Startup Delay**, and *(v)* **Average P1203 MOS**.

Table 3.9: Experimental Parameters Usage Based on Different Combinations

Experiment No	Cellular Network	Total Traces	DASH Client	ABS Algorithms	Transport Option	Video Duration	Cross Traffic	Experiment Repetition
1	3G 4G 5G	15	1 (Single) 3 (Parallel)	Conventional BBA Arbiter	TCP & QUIC	3 min	W/ (UDP) & W/O W/O	3 times
2	5G	1	1 (Single)	Conventional BBA Arbiter	TCP & QUIC	3 min	W/ (UDP) & W/O	3 times

3.4 Concluding Remarks

The overall design requirement regarding `Edge QoE Probe` was described for QoE prediction at the edge of the network and QoE performance evaluation over different transport protocols. For implementation, an emulation-based testbed was developed to stream DASH video content by replaying different cellular network traces bandwidth. Network traffic as QoS format and video log as ground truth were used to train different supervised ML models for QoE KPIs prediction. Moreover, we performed a QoE performance evaluation of DASH video over TCP and QUIC transport. The experimental results and analysis are presented in the next chapter.

Chapter 4

Experimental Evaluation Analysis

This chapter evaluates the supervised ML-based QoE estimation performance of the proposed `Edge QoE Probe` for end-to-end encrypted DASH video service traffic. Moreover, a detailed discussion of the results obtained from the experiment stated in the previous chapter about DASH-supported video streaming QoE performance over TCP and QUIC transport protocol for different ABS algorithms and diverse network conditions are provided in this chapter.

4.1 QoE Prediction at Edge Premises

This section evaluates the performance of the supervised ML model built using separately generated datasets on the TCP and QUIC transport protocols for the DASH video service from the controlled experiment. This work aimed to estimate the QoE of the DASH video service using the QoS measurement capabilities at the network level extracted from the network's edge. Thus, the overall QoE prediction was divided into two parts: real-time QoE prediction, and per-session QoE prediction.

We considered only two QoE KPIs, namely resolution and bit-rate, for real-time (0.5-second interval) prediction. In contrast, for per-session, we considered five QoE KPIs, namely resolution, bit-rate, quality switches, stall ratio, startup delay, and one QoE model output, namely P1203 MOS.

We picked seven supervised ML algorithms for the classification task and six supervised ML algorithms for the regression task. They were- Linear Regression (LR^r), Logistic Regression (LR^c), K-Nearest Neighbors Classifier (KNC), K-Nearest Neighbors Regressor (KNR), Gaussian Naive Bayes (GNB), Support Vector Machine Classifier (SVC), Support Vector Machine Regressor (SVR), Decision Tree Classifier (DTC), Decision Tree Regressor (DTR), Random Forest Classifier (RFC), Random Forest Regressor (RFR), Multi-layer Perceptron Classifier (MPC), and Multi-layer Perceptron Regressor (MPR).

The well-known python-based scikit-learn[1] library was used for implementing each ML model (including a Multi-layer Perceptron Neural Network). We evaluated each ML model's performance on a system with 10-core, 2.20 GHz CPUs (Intel Xeon(R) Silver 4114 CPU) and 64 GB of RAM.

In both real-time and per-session QoE prediction, we performed model benchmarking. For this purpose, we compared seven (for classification task) and six (for regression task) ML algorithms models and selected the best model based on cross-validation reports. To select the best model, we used the entire dataset to train/build models. Firstly, the hyper-parameters

[1]https://scikit-learn.org/stable/

were chosen for each model using an exhaustive grid search[2] and selected the best parameters that maximize the scores (e.g., mean accuracy, R^2) and avoid over-fitting.

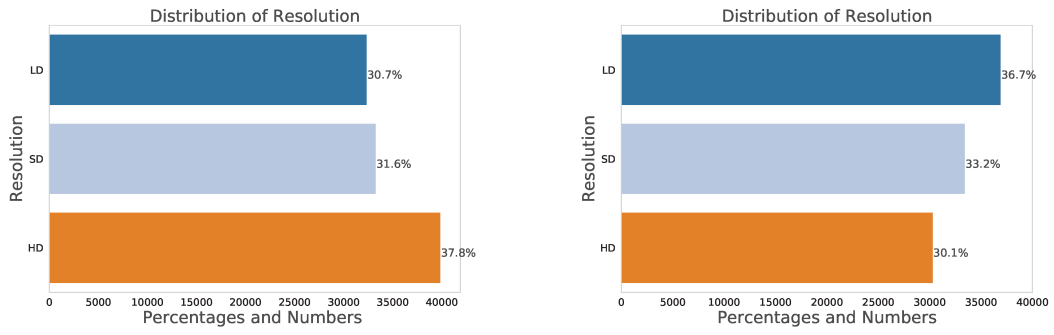
After tuning the hyper-parameters, a 5-fold cross-validation method was used to compare the overall scores and training time (e.g., the time for fitting the estimator on the train set for each cross-validation split) among different models. During the training of each model, cross-validation was used on the training portion dataset. In 5-fold cross-validation, the training portion dataset was split into a 5 number of sections/folds where a portion of the fold was used as a validation set, and the remaining part was used as the training set. Thus, each of the models calculated scores (e.g., accuracy, R^2) and training time five times and, lastly, computed average scores (e.g., accuracy, R^2) and training time. Then, by getting the most suitable algorithm from the model benchmark, further evaluation such as feature importance/selection and reduced features results were conducted on the selected suitable algorithm's model using the same tuned hyper-parameters.

A detailed discussion of the results on ML-assisted real-time and per-session QoE prediction is stated below.

4.1.1 Real-time QoE Prediction

The datasets generated over TCP and QUIC transport per time interval were used to make the ML model included temporal QoS metrics as a feature and resolution and bit-rate QoE KPIs from customized video logs as a ground truth. Thus, both QoE KPIs per time interval was considered to use as a target to build ML models. This approach allows for obtaining real-time predictions for any video session. Specifically, it would be feasible to forecast the resolution and bit-rate outcomes for each current time interval. The dataset contains a total of 105613 entries over TCP and 100757 entries over QUIC for each 0.5 second time interval of video playback.

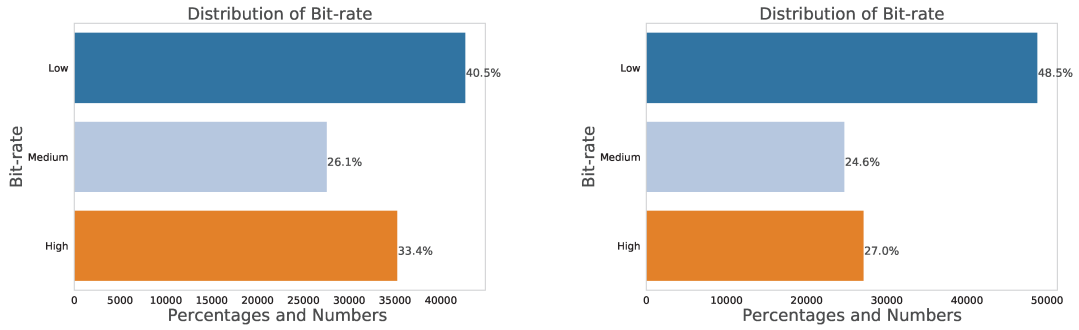
As stated earlier, firstly, we benchmarked different ML algorithms using the entire dataset to train the models and select the best algorithm based on cross-validation results that suit our real-time QoE KPIs prediction. Next, the entire datasets were divided into multiple sets of a new dataset based on all and reduced QoS features computed on different time windows. Then, each dataset was split into training and testing in the ratio of 80% and 20%. Lastly, we used the training portion dataset to train the model over the obtained best algorithm and evaluated its performance based on the testing dataset. We used the classification approach for the following QoE KPIs.



(a) (b)

Figure 4.1: Distribution of Resolution Classes across (a) TCP and (b) QUIC Datasets

[2]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html



(a) (b)

Figure 4.2: Distribution of Bit-rate Classes across (a) TCP and (b) QUIC Datasets

- **Resolution**: To estimate real-time resolution KPI, we used the classification method with three classes: LD, SD, and HD. Class "LD" or Low Definition denotes that video was played in 320x180 and 375x216 resolution. Likewise, class "SD" or Standard Definition means that video was played in 512x288, 640x360, and 736x414 resolution. Lastly, class "HD" or High Definition corresponds to 1280x720 and 1920x1080 resolution.

  Figure 4.1 depicts the distributions of resolution classes across the real-time-based TCP and QUIC datasets. The majority of resolution instances belong to the HD class (37.8%) over TCP and LD class (36.7%) over QUIC datasets. Thus, the distribution of resolution classes indicates all the ABS algorithms for DASH service were aggressive to hold higher resolution over TCP transport.

- **Bit-rate**: To estimate real-time bit-rate KPI, we classified bit-rate into three classes: Low, Medium, and High. Where class "Low" denotes that the video was played in less than 700 Kbps bit-rate. Likewise, class "Medium" corresponds to above 700 Kbps and below 2500 Kbps bit-rate, and "High" corresponds to above 2500 Kbps.

  Figure 4.2 depicts the distributions of bit-rate classes across the real-time-based TCP and QUIC datasets. We observe that Low class over the TCP dataset (40.5%) and the QUIC dataset (48.5%) dominated most. However, the instances of High class over the TCP dataset were higher than QUIC.

Model benchmarking. We evaluated seven multi-class classifiers' performance over the TCP and QUIC dataset separately. The tuned parameters used to train each classifier for both resolution and bit-rate KPI predictions are given in Table 4.1. Moreover, Table 4.2 shows each multi-class classifier's 5-fold cross-validation average accuracy and training time.

We note that the GNB achieved the least amount of time for training but provided the worst accuracy. The SVC took the highest amount of time for the training and exhibited second-worst accuracy. The MPC showed the highest variability in the case of accuracy.

In general, out of the seven classifier's DTC and RFC achieved higher accuracy, nearly 90% over the TCP dataset and above 90% over the QUIC dataset. Though RFC took slightly high training time than DTC, we assume it is a reasonable time for training, and RFC achieved approximately 3% improvement in the accuracy compared to the DTC. Based on the obtained results, we conclude that the Random Forest Classifier (RFC) was the most appropriate for real-time resolution and bit-rate prediction tasks. For further evaluation, we use RFC with the same tuned parameters.

Table 4.1: ML Model with Tuned Hyper-parameter for Real-time Two KPIs Prediction

ML Model	**QoE KPIs**	**Tuned Parameters**	
		TCP Dataset	QUIC Dataset
LR^c	Resolution	penalty='l2', solver='lbfgs'	penalty='l2', solver='lbfgs'
	Bit-rate	penalty='l2', solver='lbfgs'	penalty='l2', solver='lbfgs'
KNC	Resolution	n_neighbors=7	n_neighbors=3
	Bit-rate	n_neighbors=7	n_neighbors=3
GNB	Resolution	priors=None, var_smoothing=1e-09	priors=None, var_smoothing=1e-09
	Bit-rate	priors=None, var_smoothing=1e-09	priors=None, var_smoothing=1e-09
SVC	Resolution	C=1, kernel='poly', gamma='scale'	C=1, kernel='poly', gamma='scale'
	Bit-rate	C=1, kernel='poly', gamma='scale'	C=1, kernel='poly', gamma='scale'
DTC	Resolution	criterion='entropy', max_depth=None, class_weight='balanced', min_sample_leaf=3, min_samples_split=4	criterion='entropy', max_depth=50 class_weight='balanced', min_sample_leaf=5, min_samples_split=2
	Bit-rate	criterion='entropy', max_depth=50, class_weight=None, min_sample_leaf=3, min_samples_split=2	criterion='entropy', max_depth=50 class_weight=None, min_sample_leaf=5, min_samples_split=2
RFC	Resolution	criterion='gini', max_depth=90, class_weight='None', min_sample_leaf=4, min_samples_split=12, n_estimators=500	criterion='gini', max_depth=100 class_weight='balanced', min_sample_leaf=3, min_samples_split=10, n_estimators=500
	Bit-rate	criterion='gini', max_depth=50, class_weight='balanced', min_sample_leaf=3, min_samples_split=10, n_estimators=200	criterion='gini', max_depth=50 class_weight='balanced', min_sample_leaf=3, min_samples_split=10, n_estimators=500
MPC	Resolution	solver='adam', activation='identity', hidden_laye_sizes=(100,), alpha=0.05	solver='adam', activation='identity', hidden_layer_sizes=(50, 100, 50), alpha=0.0001
	Bit-rate	solver='adam', activation='identity', hidden_laye_sizes=(50, 100, 50), alpha=0.05	solver='adam', activation='identity', hidden_layer_sizes=(50, 50, 50), alpha=0.0001

Table 4.2: Benchmarking of Seven Supervised ML Model for Real-time Two KPIs Prediction

ML Model	**QoE KPIs**	**Accuracy (%)**		**Training Time (min)**	
		TCP Dataset	QUIC Dataset	TCP Dataset	QUIC Dataset
LR^c	Resolution	67 (± 0.91)	65 (± 1.43)	0.07 (± 0.002)	0.06 (± 0.003)
	Bit-rate	67 (± 0.68)	64 (± 1.11)	0.08 (± 0.002)	0.06 (± 0.004)
KNC	Resolution	79 (± 1.02)	82 (± 1.00)	0.09 (± 0.001)	0.05 (± 0.001)
	Bit-rate	78 (± 0.63)	82 (± 0.78)	0.09 (± 0.001)	0.05 (± 0.001)
GNB	Resolution	56 (± 2.57)	62 (± 0.88)	0.008 (± 0.00009)	0.005 (± 0.0002)
	Bit-rate	55 (± 1.02)	56 (± 0.97)	0.008 (± 0.00005)	0.005 (± 0.0001)
SVC	Resolution	64 (± 0.91)	57 (± 0.52)	39.01 (± 0.239)	27.62 (± 0.240)
	Bit-rate	64 (± 0.73)	58 (± 0.53)	40.66 (± 0.137)	26.56 (± 0.132)
DTC	Resolution	88 (± 1.03)	92 (± 0.46)	0.20 (± 0.004)	0.18 (± 0.007)
	Bit-rate	88 (± 0.80)	94 (± 0.98)	0.19 (± 0.003)	0.16 (± 0.005)
RFC	Resolution	91 (± 0.28)	96 (± 0.54)	3.41 (± 0.014)	3.36 (± 0.023)
	Bit-rate	91 (± 0.44)	96 (± 0.55)	1.34 (± 0.022)	3.14 (± 0.020)
MPC	Resolution	68 (± 5.19)	64 (± 7.33)	2.16 (± 0.020)	2.62 (± 0.027)
	Bit-rate	72 (± 1.16)	69 (± 5.79)	2.93 (± 0.031)	2.51 (± 0.044)

Feature importance. Then, we analyzed different feature influences or importance in terms of resolution and bit-rate predictions. For this purpose, we divided the full feature (TCP-252 features, QUIC-168 features) of the dataset into a total of six feature subsets with two categories. The first category contains three subsets of features from the *current* window (TCP-36 features, QUIC-24 features), *trend* window (TCP-180 features, QUIC-120 features), and *session* window (TCP-36 features, QUIC-24 features). In these subsets, all the features computed in each particular window were considered.

For the second category, we leveraged the features relative importance score to select the top 15 features from each window using a multi-class RFC with a fixed seed for controlling randomness. After that, we split each feature subsets into the ratio of 80% (training) and 20% (testing). We used the RFC to build a model with a training portion of data and evaluated the precision, recall, and F1-score of the model based on testing data from each of these six feature subsets. The model prediction performance reports for each feature subsets are given in Tables 4.3 and 4.4.

Takeaway: The analysis indicates that the top 15 feature subsets provided similar performance as full feature subsets provided. Figure 4.3 and 4.4 exhibits the up/downlink packets inter-arrival time QoS KPI for the *current*, the uplink packet size QoS KPI for the *trend*, and the downlink average throughput QoS KPI for *session* window hold high relative importance

Table 4.3: Different Feature Subsets' Model Performance for Real-time Resolution KPI

Feature Set	**Class**	**TCP Dataset**			**QUIC Dataset**		
		Precision	Recall	F1-score	Precision	Recall	F1-score
Current_{full}	LD	77%	41%	54%	79%	53%	63%
	SD	70%	48%	57%	76%	60%	67%
	HD	56%	91%	69%	54%	90%	68%
	Weighted Average	**67%**	**62%**	**61%**	**71%**	**66%**	**66%**
$\text{Current}_{top\text{-}15}$	LD	78%	41%	54%	90%	53%	63%
	SD	71%	48%	57%	77%	60%	68%
	HD	56%	92%	69%	54%	90%	68%
	Weighted Average	**67%**	**63%**	**61%**	**71%**	**67%**	**66%**
Trend_{full}	LD	97%	77%	86%	97%	83%	90%
	SD	97%	85%	90%	97%	89%	93%
	HD	79%	99%	88%	78%	99%	87%
	Weighted Average	**90%**	**88%**	**88%**	**91%**	**90%**	**90%**
$\text{Trend}_{top\text{-}15}$	LD	94%	76%	84%	97%	83%	89%
	SD	94%	83%	88%	96%	89%	92%
	HD	78%	98%	87%	77%	98%	87%
	Weighted Average	**88%**	**87%**	**87%**	**91%**	**89%**	**89%**
Session_{full}	LD	99%	98%	98%	99%	99%	99%
	SD	98%	97%	97%	99%	98%	98%
	HD	98%	100%	99%	99%	100%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**
$\text{Session}_{top\text{-}15}$	LD	98%	97%	98%	99%	99%	99%
	SD	98%	97%	97%	99%	98%	98%
	HD	98%	99%	99%	99%	100%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**
$\text{All-window}_{top\text{-}15}$	LD	98%	97%	98%	99%	99%	99%
	SD	98%	97%	97%	98%	98%	98%
	HD	98%	99%	99%	99%	100%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**

Table 4.4: Different Feature Subsets' Model Performance for Real-time Bit-rate KPI

Feature Set	**Class**	**TCP Dataset**			**QUIC Dataset**		
		Precision	Recall	F1-score	Precision	Recall	F1-score
Current_{full}	Low	82%	48%	60%	86%	56%	68%
	Medium	67%	44%	53%	67%	58%	62%
	High	52%	91%	66%	50%	89%	64%
	Weighted Average	**68%**	**61%**	**60%**	**72%**	**65%**	**65%**
$\text{Current}_{top\text{-}15}$	Low	82%	47%	60%	87%	56%	68%
	Medium	68%	44%	54%	69%	60%	64%
	High	52%	91%	66%	51%	89%	65%
	Weighted Average	**68%**	**61%**	**60%**	**73%**	**66%**	**66%**
Trend_{full}	Low	97%	81%	88%	99%	86%	92%
	Medium	96%	82%	88%	97%	88%	92%
	High	75%	99%	85%	74%	99%	85%
	Weighted Average	**89%**	**87%**	**87%**	**92%**	**90%**	**90%**
$\text{Trend}_{top\text{-}15}$	Low	96%	80%	87%	98%	85%	91%
	Medium	93%	80%	86%	95%	88%	91%
	High	75%	98%	85%	74%	98%	85%
	Weighted Average	**88%**	**86%**	**86%**	**91%**	**89%**	**89%**
Session_{full}	Low	99%	98%	98%	100%	99%	99%
	Medium	97%	95%	96%	99%	99%	98%
	High	97%	99%	98%	99%	99%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**
$\text{Session}_{top\text{-}15}$	Low	98%	98%	98%	100%	99%	99%
	Medium	97%	95%	96%	98%	98%	98%
	High	97%	99%	98%	99%	99%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**
$\text{All-window}_{top\text{-}15}$	Low	99%	98%	98%	100%	99%	99%
	Medium	97%	95%	96%	98%	98%	98%
	High	97%	99%	98%	99%	99%	99%
	Weighted Average	**98%**	**98%**	**98%**	**99%**	**99%**	**99%**

for both QoE KPIs prediction.

We observe that *current* and *trend* window features are not adequate to predict video QoE KPIs according to recall, precision, and F1-score. For the *trend* window, we found that the larger time duration interval's features (e.g., 20 and 10 second) had the most influence on prediction.

The model with only the *session* window's features yielded (98% over TCP and 99% over QUIC dataset) the highest accuracy for both QoE KPIs prediction. Moreover, it is noticeable that most of the top 15 features from all window's full feature datasets belong to the *session* window and provided similar results.

Figure 4.3: Top 15 Important Features for Resolution Prediction: Current Window- (a) TCP (b) QUIC, Trend Window- (c) TCP (d) QUIC, Session Window- (e) TCP (f) QUIC, All-Window- (g) TCP (h) QUIC Datasets

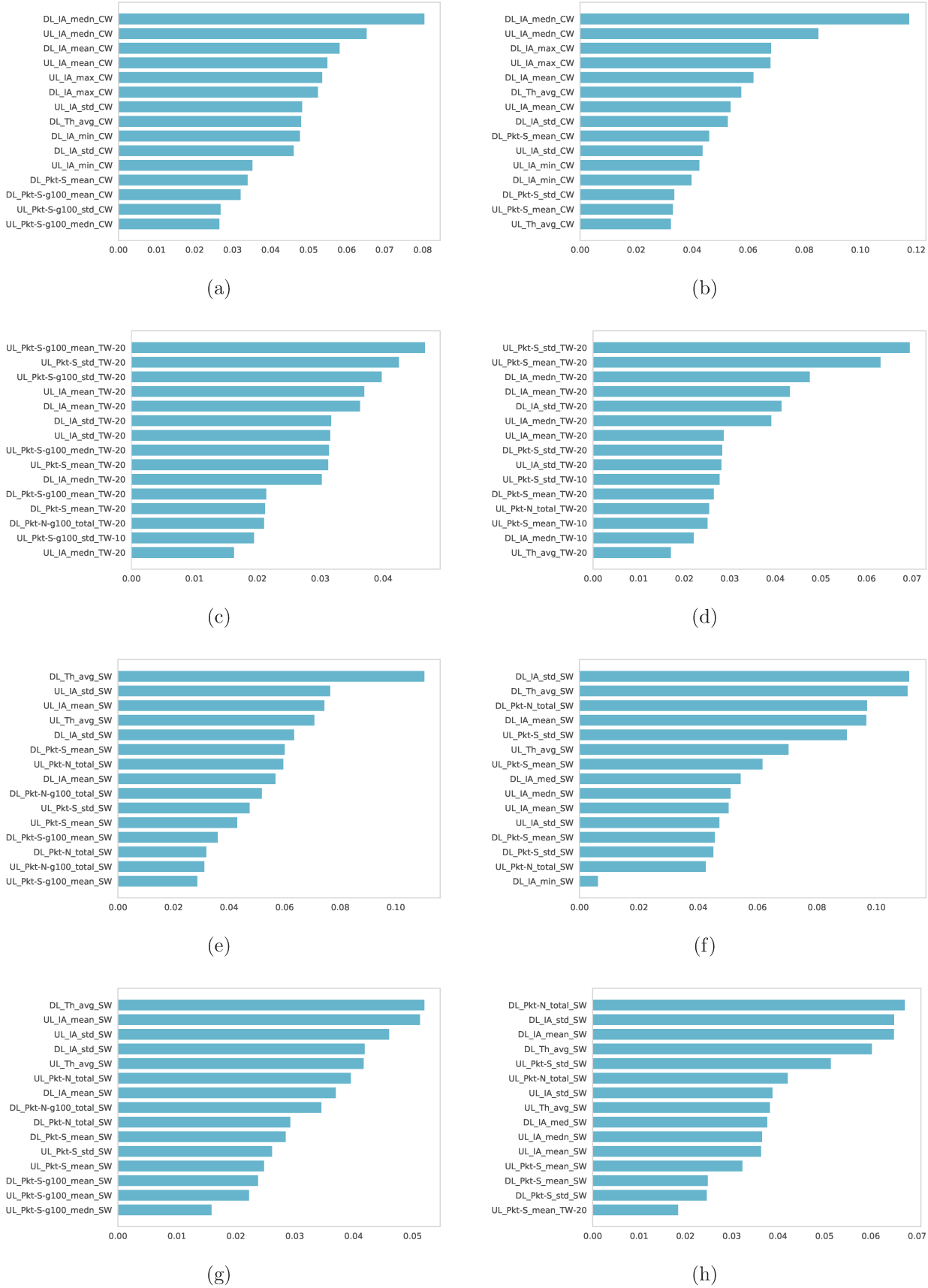


Figure 4.4: Top 15 Important Features for Bit-rate Prediction: Current Window- (a) TCP (b) QUIC, Trend Window- (c) TCP (d) QUIC, Session Window- (e) TCP (f) QUIC, All-Window- (g) TCP (h) QUIC Datasets

4.1.2 Per-session QoE Prediction

The dataset generated over TCP and QUIC transport separately to estimate per session QoE as well. The aggregated QoS KPIs from the *current* window as feature input and goDASH provided the entire session's outputs as ground truth were used to train the ML models for estimating per session QoE KPIs. We considered a total of six KPIs for per-session QoE estimation, namely average resolution, average bit-rate, quality switches, stall ratio, startup delay, and average MOS based on ITU-T Rec. P.1203.1 model. For the following five QoE KPIs, we used the classification approach.

- **Average Resolution**: The average resolution was labeled in three classes. Resolution 320x180, and 375x216 as Low Definition (LD), 512x288, 640x360, and 736x414 as Standard Definition (SD) and 1280x720 and 1920x1080 as High Definition (HD).
- **Average Bit-rate**: The average bit-rate was labeled in three classes. Bit-rate less than 700 Kbps as Low, between 700 and 2500 Kbps as Medium, and above 2500 Kbps as High.
- **Quality Switches**: As opposed to coarse-grained binary classification (video being played with the consistent quality or quality changes at least once) stated in the work [54], we used a multi-class approach for quality switches. We defined quality switches 0 to 5 times as Low, 6 to 10 times as Medium, and above 10 times as High.
- **Stall Ratio**: Based on a similar approach in work [33] stall ratio was labeled in three classes. Ratio value 0 as No Stall, 0 to 0.1 as Mild Stall, and above 0.1 as Severe Stall.
- **Average MOS**: To ease classification, MOS was labeled in three classes as well. MOS values 1 to 2.5 as Poor, 2.6 to 3.5 as Fair, and above 3.5 as Good.

We observe the overall per video session datasets from the distribution Figures 4.5 - 4.9 were not big enough, and for a few QoE KIPs' datasets were highly imbalanced. Hence, before train ML classifiers for each QoE KPI, we handled the data imbalances by balancing the number of instances or entries per class with upsampling the minority classes using SMOTE[3] (Synthetic Minority Oversampling Technique) to avoid bias models.

Moreover, we used the ML regression method to estimate the following two QoE KPIs.

- **Startup Delay**: As opposed to previous work [55] to classify (binary classification) the video started or not based on startup delay threshold, we tried to predict more fine-grained startup delay time using ML regressor. We observe from Figure 4.10 that most of the video sessions over our generated TCP and QUIC datasets took around 0.5 to 1 second to start the playback.
- **Average MOS**: Alongside the classification approach to estimate MOS, we also used the regression approach to achieve finer prediction granularity of MOS since our generated dataset contains the continuous values of MOS range between 0 to 5. The distribution Figure 4.11 shows that the MOS over the TCP dataset calculated between 1.5 to 3.5 values and over the QUIC dataset calculated between 1.5 to 4 values, but values around 1.75 to 2 contain high-frequency density.

[3]https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html

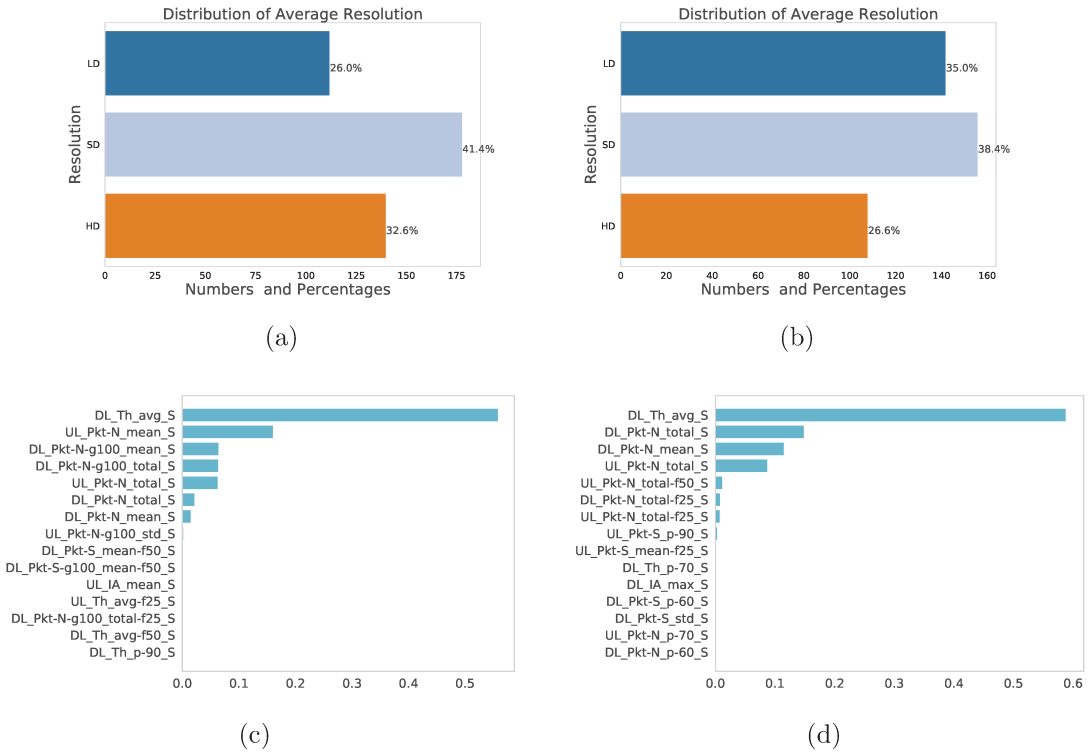


Figure 4.5: Distribution of Per-session Resolution Classes across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

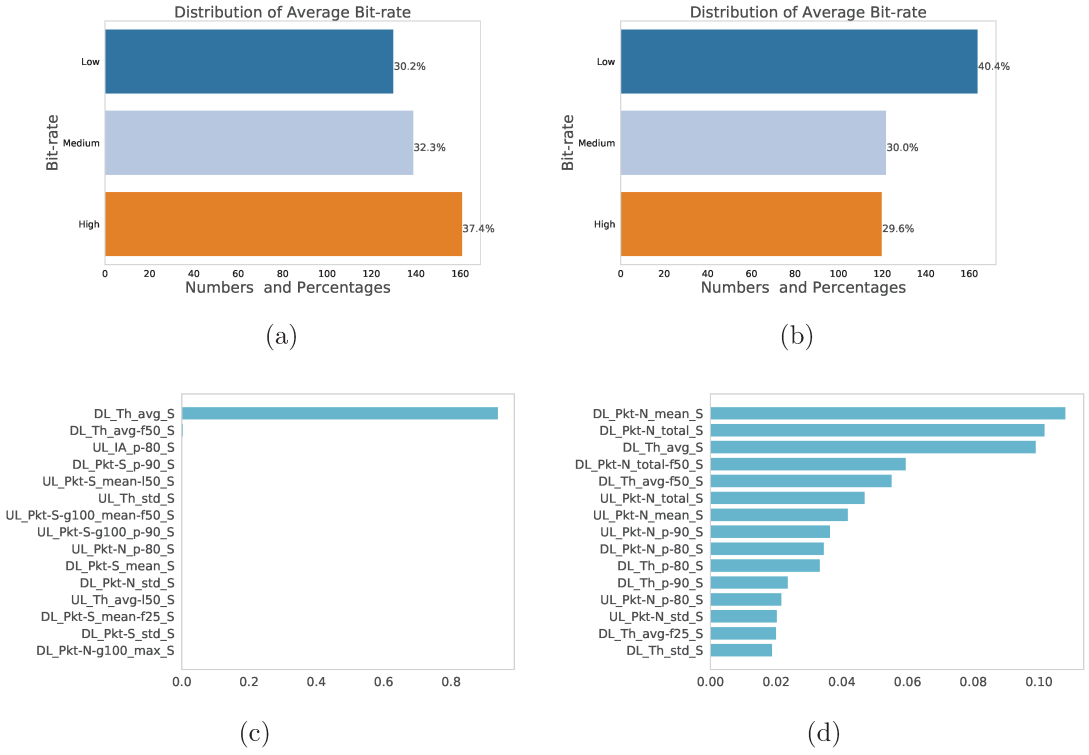


Figure 4.6: Distribution of Per-session Bit-rate Classes across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

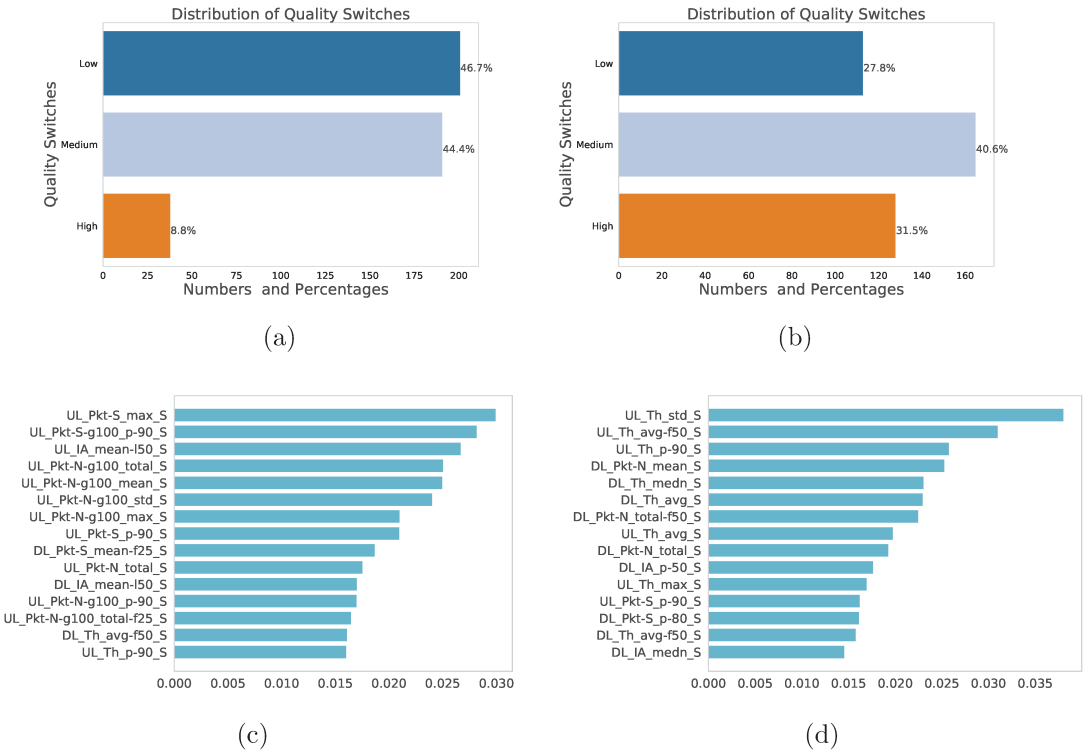


Figure 4.7: Distribution of Per-session Quality Switches Classes across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

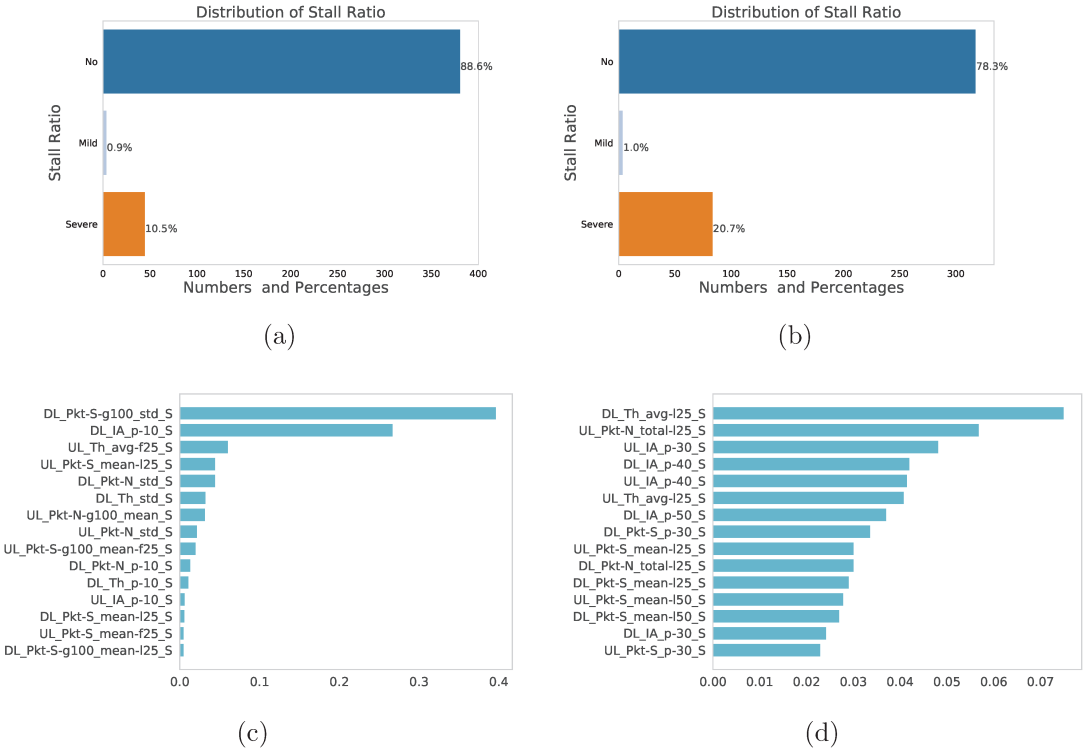


Figure 4.8: Distribution of Per-session Stall Ratio Classes across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

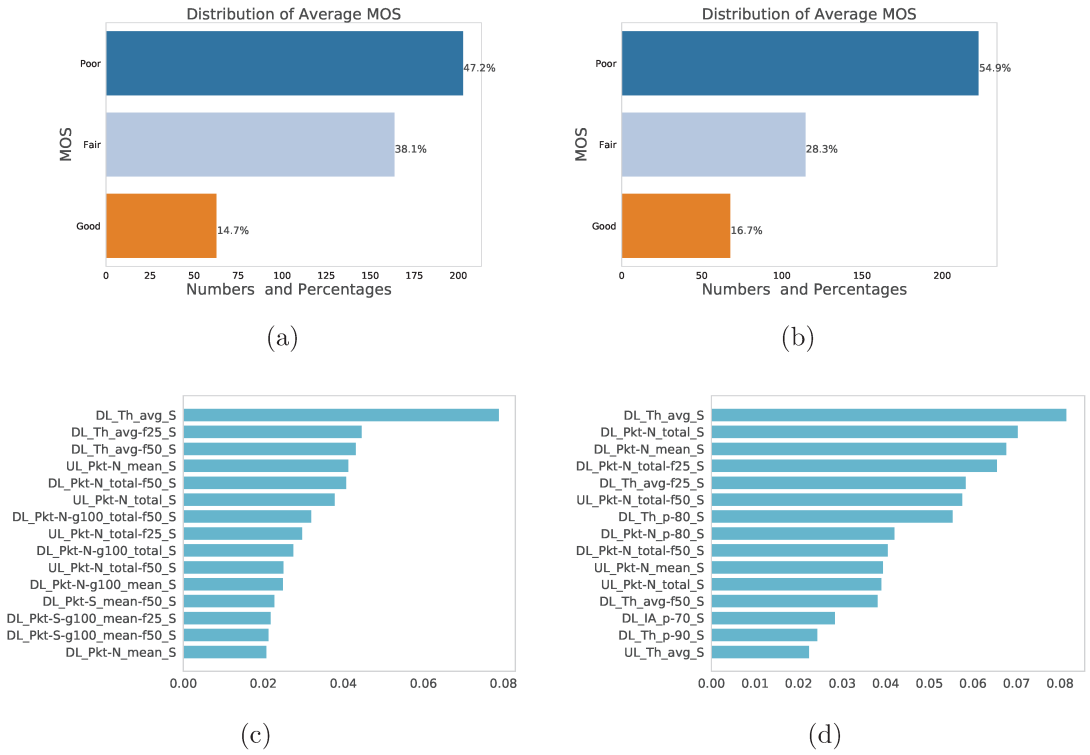


Figure 4.9: Distribution of Per-session MOS Classes across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

Model benchmarking. We used both the classification and regression method for per-session QoE KPIs estimation. Therefore, we evaluated both classifiers' and regressors' performance over TCP and QUIC datasets. We tuned the parameters to train each classifier and regressor as well, maximizing the accuracy (for classification) and R^2 score (for regression). Tables 4.5 and 4.6 contain average accuracy and R^2 score from the 5-fold cross-validation. Due to the small size of the dataset, all the classifiers and regressors took less than 1 minute to train. Hence, we omitted the training time from Tables 4.5 and 4.6.

We note that RFC among all the classifiers presented the highest average accuracy for each QoE KPI. Then, DTC provided the second-highest average accuracy. However, SVM as a classifier showed the worst performance in terms of per-session QoE KPIs estimation.

In contrast, regression analysis for startup delay and MOS QoE KPIs, RFR showed the maximum R^2 score, which indicates that data over the RFR model mostly close to the fitted regression line. Both LR^r (Linear Regression) and MPR showed the negative R^2 score, which means the models did not follow the data trend, so it fits the regression line worse than a horizontal line using the mean value.

Consequently, we conclude that Random Forest acts as the most suitable algorithm to estimate per-session QoE KPIs for classification and regression tasks. Further evaluation conducts using only the Random Forest ML algorithm.

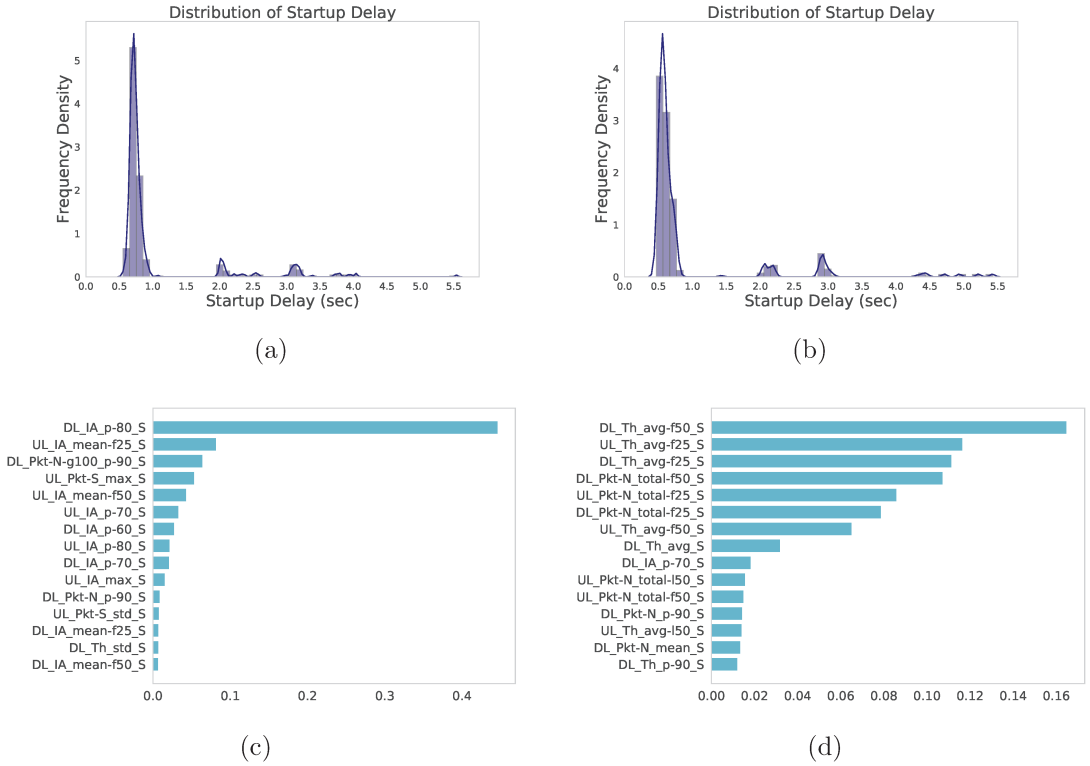


Figure 4.10: Distribution of Per-session Startup Delay Values across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

Feature importance. To more analyze the RFC (classification) and RFR (regression) performance with the relative feature importance, we split the dataset in the ratio of 67% (training) and 33% (testing). Note that, for the classification task, we balanced all class instances or entries. We trained the RFC and RFR using the same tuned parameters obtained from grid search carried out during per-session model benchmarking. Based on testing data classification report (e.g., precision, recall, and F1-score) for five QoE KPIs and regression result (e.g., R^2 score, and RMSE) for two QoE KPIs are presented in Table 4.7 and 4.8. We also evaluated the most influential QoS features for the trained model (Random Forest-based) that play a vital role in estimating per-session QoE KPIs.

Takeaway: The key finding to estimate each of the QoE KPIs with QoS features importance are given as follows.

- *Average Resolution*: The model accurately predicted the LD and SD resolution classes (recall 100%) over the TCP dataset and LD and HD resolution classes (recall 100%) over the QUIC dataset. Figure 4.5 (c) (d) depicts RFC-based resolution classification over both TCP and QUIC dataset strongly relies on the entire session's downlink average throughput QoS feature.

- *Average Bit-rate*: The model only showed 6% (recall 94%) incorrect predictions over the QUIC dataset for the High bit-rate class. Figure 4.6 (c) (d) depicts RFC-based bit-rate classification strongly relies on the entire session's downlink average throughput QoS feature only over the TCP dataset. On the other hand, the first half and the entire session's downlink average throughput and the number of packets QoS feature mostly influence bit-rate classification over the QUIC dataset.

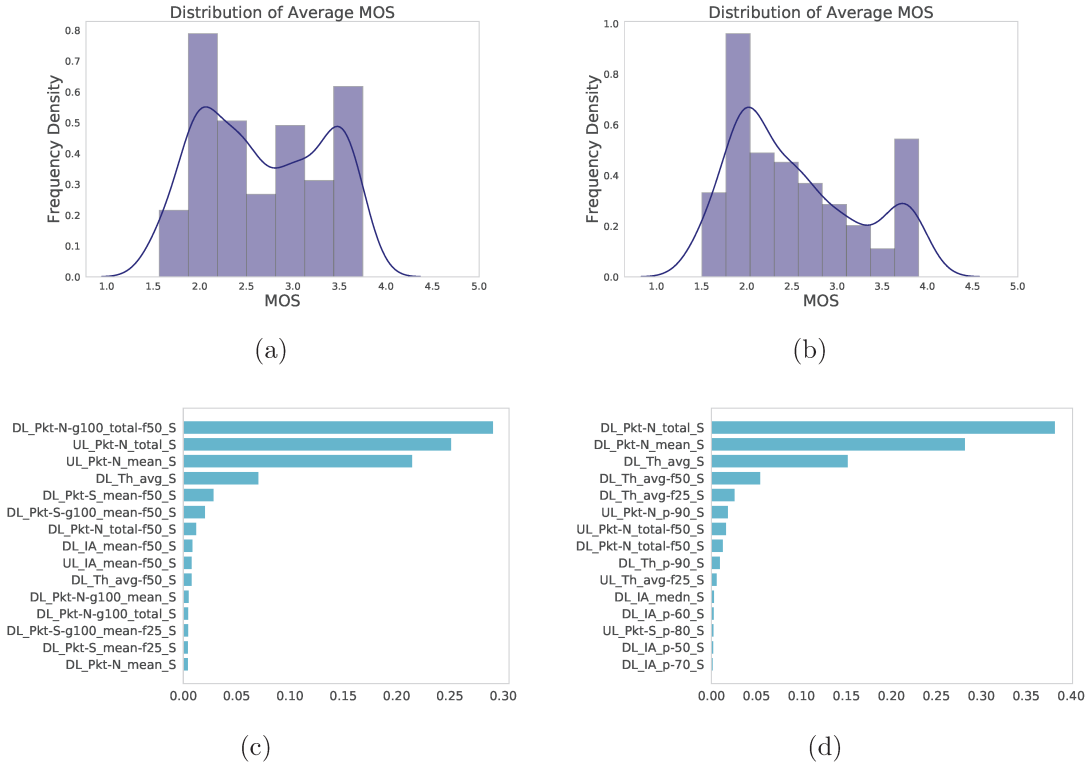


Figure 4.11: Distribution of Per-session MOS Values across (a) TCP and (b) QUIC Datasets and Top 15 Important Features across (c) TCP and (d) QUIC Datasets

- *Quality Switches:* According to the weighted average[4] of recall, the overall model 14-13% incorrectly predicted quality switches classes. Uplink packet size, the number of packets, packets inter-arrival time QoS features over TCP dataset (Figure 4.7 (c)), and uplink throughput QoS features over QUIC dataset (Figure 4.7 (d)) mostly influence predict quality switches.

- *Stall Ratio*: The classifier showed the most incorrect prediction for only the No-Stall class. Downlink packet size and up/downlink 10th percentile QoS features holds more crucial influence for stall ratio prediction over TCP dataset (Figure 4.8 (c)). Over the QUIC dataset (Figure 4.8 (d)), the last quarter of the session's QoS features (up/downlink average throughput, the number of packets) and 30-50th percentile QoS features (uplink packets inter-arrival time, packet size) mostly influence to predict stall ratio.

- *Average MOS*: For classification, the model over TCP dataset 3% (97% weighted average of recall) and QUIC dataset 1% (99% weighted average of recall) incorrectly predicted MOS classes. The first quarter, half, and entire session's downlink average throughput and the number of packets QoS features strongly influence to predict MOS over both datasets (Figure 4.9 (c) (d)). For regression, the model showed the 98% R^2 score over both datasets, which indicates data were mostly close to the fitted regression line; thus, we observe negligible RMSE (Table 4.8). Downlink the number of packets and average throughput QoS features (Figure 4.11 (c) (d)) comparatively play the most crucial role in MOS regression analysis.

[4]https://en.wikipedia.org/wiki/Weighted_arithmetic_mean

- *Startup Delay*: The regression model obtained a reasonable R^2 score (more than 80%) to follow the data trend over both datasets. Up/downlink packets inter-arrival time QoS features specifically for the high percentile (80th) and the first quarter of the session over TCP dataset (Figure 4.10 (c)) and up/downlink average throughput QoS features for the first quarter and half of the session over QUIC dataset (Figure 4.10 (d)) mostly influence to predict startup delay. In general, we conclude that the first quarter and half of the session's QoS features strongly influence startup delay.

Table 4.5: Benchmarking of Seven Supervised ML Model for Per-session Five KPIs Prediction

ML Model	**QoE KPIs**	**Accuracy (%)**	
		TCP Dataset	QUIC Dataset
LR^c	Resolution	85 (±5.46)	84 (± 4.53)
	Bit-rate	86 (± 4.54)	80 (± 5.92)
	Quality Switches	46 (± 12.27)	58 (± 5.56)
	Stall Ratio	97 (± 2.79)	94 (± 3.22)
	MOS	74 (± 10.53)	80 (± 4.59)
KNC	Resolution	75 (± 8.90)	80 (± 5.39)
	Bit-rate	80 (± 4.51)	85 (± 7.12)
	Quality Switches	76 (± 12.01)	72 (± 9.20)
	Stall Ratio	98 (± 1.52)	96 (± 4.10)
	MOS	73 (± 4.18)	85 (± 4.42)
GNB	Resolution	75 (± 6.97)	87 (± 3.51)
	Bit-rate	87 (± 2.17)	89 (± 1.88)
	Quality Switches	56 (± 5.33)	54 (± 2.30)
	Stall Ratio	92 (± 2.32)	91 (± 6.05)
	MOS	78 (± 12.50)	82 (± 10.61)
SVC	Resolution	59 (± 2.82)	51 (± 4.90)
	Bit-rate	60 (± 4.87)	47 (± 6.87)
	Quality Switches	34 (± 17.04)	48 (± 10.01)
	Stall Ratio	85 (± 7.16)	75 (± 16.73)
	MOS	40 (± 21.66)	42 (± 17.64)
DTC	Resolution	97 (± 2.60)	97 (± 2.72)
	Bit-rate	99 (± 0.41)	98 (± 1.37)
	Quality Switches	86 (± 6.77)	83 (± 5.74)
	Stall Ratio	98 (± 1.71)	97 (± 4.09)
	MOS	92 (± 1.85)	98 (± 0.87)
RFC	Resolution	98 (± 2.08)	98 (± 1.24)
	Bit-rate	99 (± 0.42)	98 (± 0.75)
	Quality Switches	89 (± 3.67)	86 (± 5.90)
	Stall Ratio	99 (± 1.92)	97 (± 2.71)
	MOS	94 (± 1.99)	98 (± 1.66)
MPC	Resolution	78 (± 10.92)	89 (± 3.25)
	Bit-rate	91 (± 4.37)	92 (± 2.83)
	Quality Switches	58 (± 10.80)	60 (± 7.54)
	Stall Ratio	99 (± 1.24)	90 (± 7.32)
	MOS	83 (± 3.46)	93 (± 6.52)

Table 4.6: Benchmarking of Six Supervised ML Model for Per-session Two KPIs Prediction

ML Model	**QoE KPIs**	**R^2 Score**	
		TCP Dataset	QUIC Dataset
LR^r	Startup Delay	-2.64 (±3.50)	$-7.3e^{+20}$ ($\pm$ $1.5e^{+21}$)
	MOS	0.28 (± 1.14)	$-2.5e^{+18}$ ($\pm$ $5.1e^{+18}$)
KNR	Startup Delay	0.73 (±0.18)	0.63 (±0.07)
	MOS	0.63 (± 0.08)	0.64 (±0.04)
SVR	Startup Delay	0.57 (±0.02)	0.04 (±0.01)
	MOS	0.51 (± 0.05)	0.20 (±0.02)
DTR	Startup Delay	0.72 (±0.14)	0.78 (±0.10)
	MOS	0.98 (± 0.01)	0.98 (±0.01)
RFR	Startup Delay	0.80 (±0.19)	0.88 (±0.02)
	MOS	0.99 (± 0.01)	0.99 (±0.001)
MPR	Startup Delay	$-2.1e^{+11}$ ($\pm$ $1.3e^{+11}$)	$-4.7e^{+11}$ ($\pm$ $3.5e^{+11}$)
	MOS	$-2.6e^{+11}$ ($\pm$ $1.5e^{+11}$	$-5.2e^{+11}$ ($\pm$ $2.3e^{+11}$)

Table 4.7: Random Forest Classifier's Report for Per-Session Five QoE KPIs

QoE KPIs	**Class**	**TCP Dataset**			**QUIC Dataset**		
		Precision	Recall	F1-score	Precision	Recall	F1-score
Resolution	LD	100%	100%	100%	100%	100%	100%
	SD	94%	100%	97%	100%	96%	98%
	HD	100%	89%	94%	97%	100%	99%
	Weighted Average	**97%**	**97%**	**97%**	**99%**	**99%**	**99%**
Bit-rate	Low	100%	100%	100%	100%	100%	100%
	Medium	100%	100%	100%	95%	100%	98%
	High	100%	100%	100%	100%	94%	97%
	Weighted Average	**100%**	**100%**	**100%**	**98%**	**98%**	**98%**
Quality Switches	Low	87%	85%	86%	79%	94%	86%
	Medium	82%	85%	83%	88%	69%	78%
	High	96%	94%	85%	92%	95%	93%
	Weighted Average	**88%**	**87%**	**87%**	**87%**	**86%**	**86%**
Stall Ratio	No-Stall	100%	99%	99%	98%	93%	96%
	Mild-Stall	99%	100%	100%	98%	100%	99%
	Severe-Stall	99%	100%	100%	96%	98%	97%
	Weighted Average	**99%**	**99%**	**99%**	**97%**	**97%**	**97%**
MOS	Poor	97%	100%	99%	100%	98%	99%
	Fair	96%	96%	96%	97%	100%	99%
	Good	98%	95%	97%	100%	100%	100%
	Weighted Average	**97%**	**97%**	**97%**	**99%**	**99%**	**99%**

Table 4.8: Random Forest Regressor's Report for Per-Session Two QoE KPIs

QoE KPIs	**TCP Dataset**		**QUIC Dataset**	
	R^2	RMSE	R^2	RMSE
Startup Delay	0.81	0.30	0.86	0.40
MOS	0.98	0.06	0.98	0.07

4.2 QoE Performance over TCP and QUIC Transport

This section shows the result obtained from two selected experiments using TCP and QUIC transport protocol separately for DASH-supported video service with different combinations, i.e., clients, ABS algorithms, and network traffic conditions.

4.2.1 Experiment 1- QoE Performance Using 3G, 4G, and 5G Network Traces for Single and Parallel DASH Clients

The entire experiment was carried out based on 15 selected network traces. We used Linux TC to change the link bandwidth after every 4 seconds so that at least two segments can easily download between 4-second intervals. This set of experiments covers the successive scenarios.

1. Single DASH client w/ and w/o background (cross) traffic to stream DASH content.
2. Three concurrent DASH clients w/o any background (cross) traffic to stream the same DASH content at the same time.

We observed negligible differences in QoE metrics' (KPIs) output w/ and w/o background (cross) traffic. Therefore, we only shed light on the result with no (w/o) background traffic for the first set of single client experiment. Figure 4.12, 4.13, and 4.14 present the overall single client video streaming performance for three ABS algorithms, three network type link utilization, and two transport protocols. Moreover, Figure 4.12, 4.13, and 4.14 contain the QoE output of video streaming performance metrics (KPIs): average bit-rate, P1203 MOS, quality switches, stall ratio, and startup delay.

Average bit-rate. The rate-based Conventional algorithm performed best to choose average bit-rate over both transport (TCP and QUIC) and three networks (3G, 4G, and 5G) scenarios. For the most part, all three ABS algorithms adopted a higher average bit-rate over TCP transport than QUIC. The average bit-rate metric results provide an interesting insight that the difference of choosing bit-rate deliberately downturn from the 3G to 5G network scenario over both TCP and QUIC. Hereby, video streaming performs nearly similar over TCP and QUIC transport when there is high bandwidth available in the link; otherwise, TCP outperforms QUIC.

P1203 MOS. The MOS results retained the same observation for the bit-rate metric. The buffer-based BBA algorithm performed poorly, and the hybrid Arbiter algorithm performed moderately in each case. However, all three ABS algorithms predominantly maintained higher MOS values over TCP.

Quality switches. Video streaming suffered the most quality changes for the BBA algorithm and the least quality changes for the Conventional algorithm. Conventional utilizes the segment quality decision based on estimated bandwidth which helps to get consistent video streaming quality. In general, the difference in quality switches over TCP and QUIC for all ABS algorithms was negligible. In the 5G network, the quality switches' amplitude was roughly equal and less than 3G and 4G network over TCP and QUIC.

Stall ratio. The obtained results show that the Conventional algorithm predominately experienced high and the BBA algorithm experienced low stall events over TCP and QUIC. We observe that Conventional always relied on high-quality segments from the earlier bit-rate and MOS results. Thus, it took more time to rebuffer while the buffer empties. On average, video streaming for all ABS algorithms suffered severe stall events on 3G and 4G networks, and the QUIC scenario showed higher stall events than TCP.

Startup delay. Interestingly, both TCP and QUIC exhibited nearly equal startup delays for each ABS algorithm. Figure 4.20 helps to interpret similar startup delay results. We noticed each ABS algorithm always downloaded video segments from the low quality over both TCP and QUIC. In this experiment set, the link's bandwidth utilization was changed after each 4-second interval which influenced the DASH player to download at least two simultaneous segments in the same bandwidth condition of 4-second interval. The first two segments always maintained the same low quality, which we set an initial buffer limit to start playback. Thus, there was no such significant difference in startup delay. The result may vary if there was a larger initial buffer threshold to start the playback. However, the 3G network held a higher average startup delay for all ABS algorithms and transports than the 4G and 5G networks.

The second set of parallel clients' experiment results is depicted in Figures 4.15, 4.16, and 4.17. In this experiment, the Conventional ABS algorithm achieved the highest bit-rate for all combinations. Similar to the single client experiment, TCP was still aggressive to download the high bit-rate than QUIC. As a consequence, each ABS algorithm predominantly gained high MOS using TCP transport. However, the behavior of the rest of the metrics of parallel clients was almost identical, likewise single client's metrics. Due to the sharing bandwidth resource for multiple clients, more stall events occurred, and overall QoE of video streaming performance was reduced than the single client as natural.

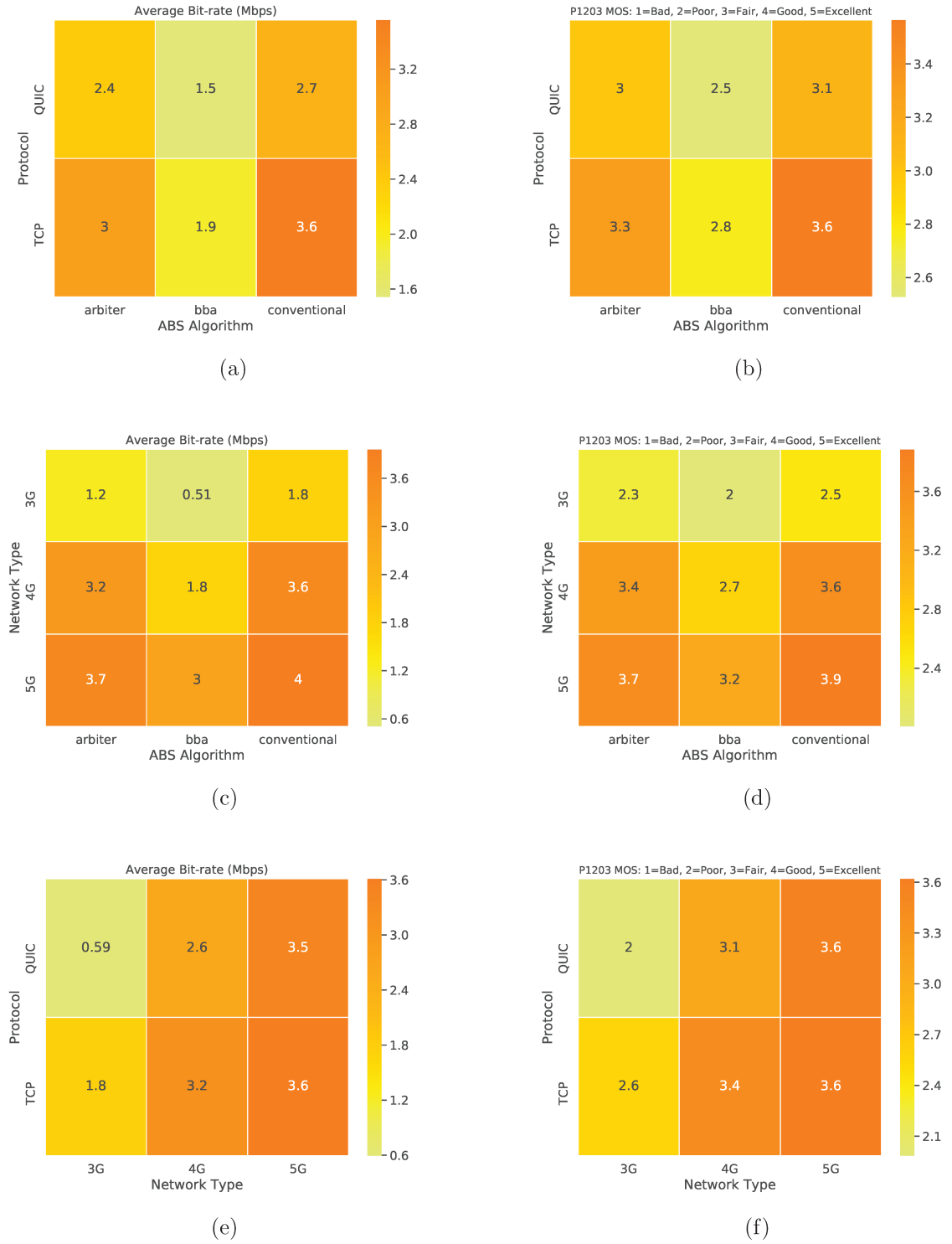


Figure 4.12: Experiment 1: Single Client- Average Bit-rate, P1203 MOS: (a) (b) Protocol vs ABS Algorithm, (c) (d) Network Type vs ABS Algorithm and (e) (f) Protocol vs Network Type

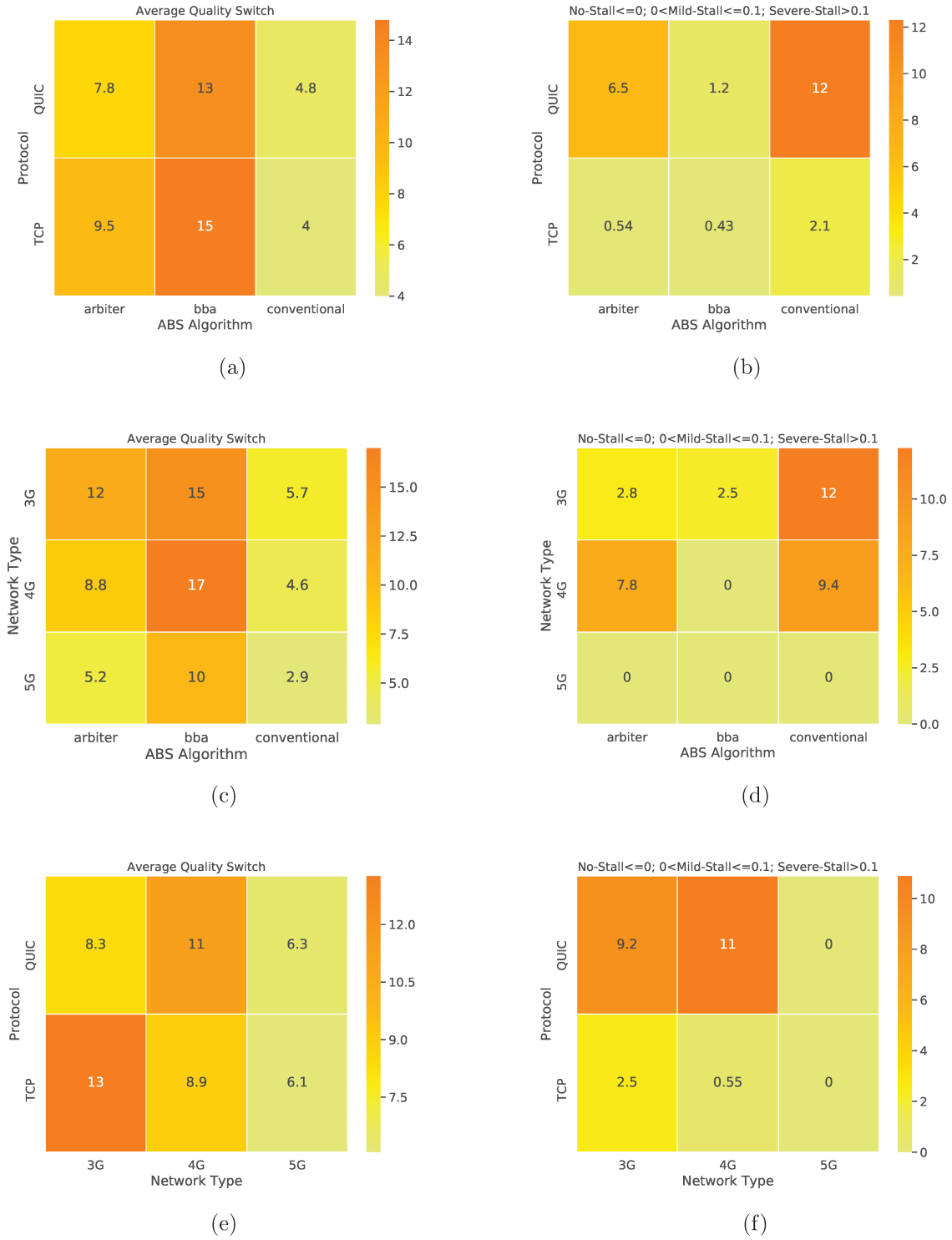
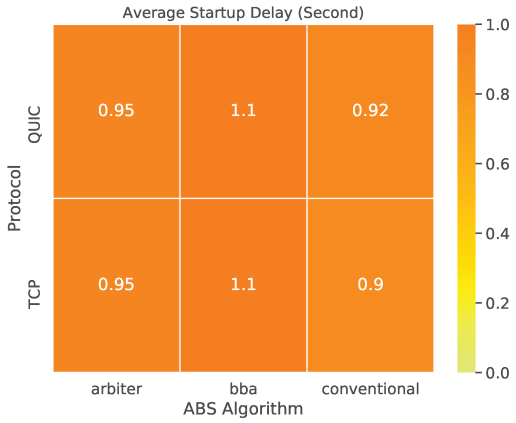
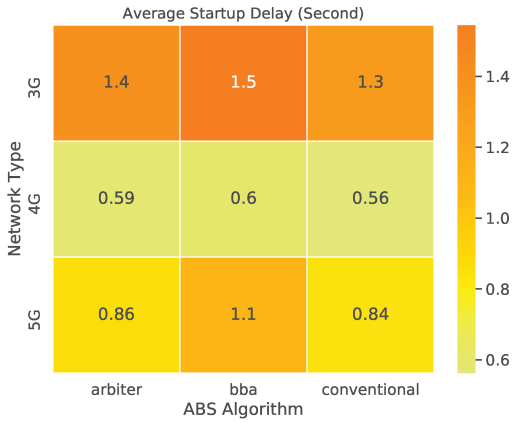


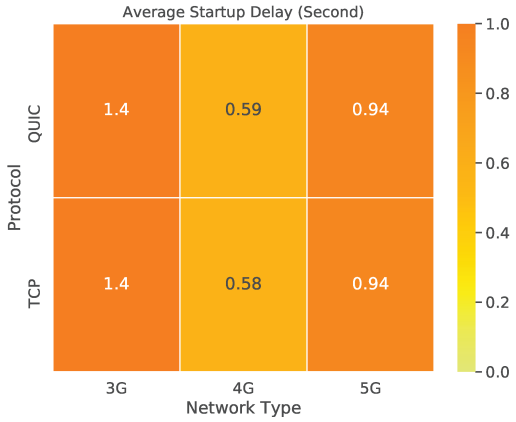
Figure 4.13: Experiment 1: Single Client- Quality Switches, Stall Ratio: (a) (b) Protocol vs ABS Algorithm, (c) (d) Network Type vs ABS Algorithm and (e) (f) Protocol vs Network Type



(a)



(b)



(c)

Figure 4.14: Experiment 1: Single Client- Startup Delay: (a) Protocol vs ABS Algorithm, (b) Network Type vs ABS Algorithm and (c) Protocol vs Network Type

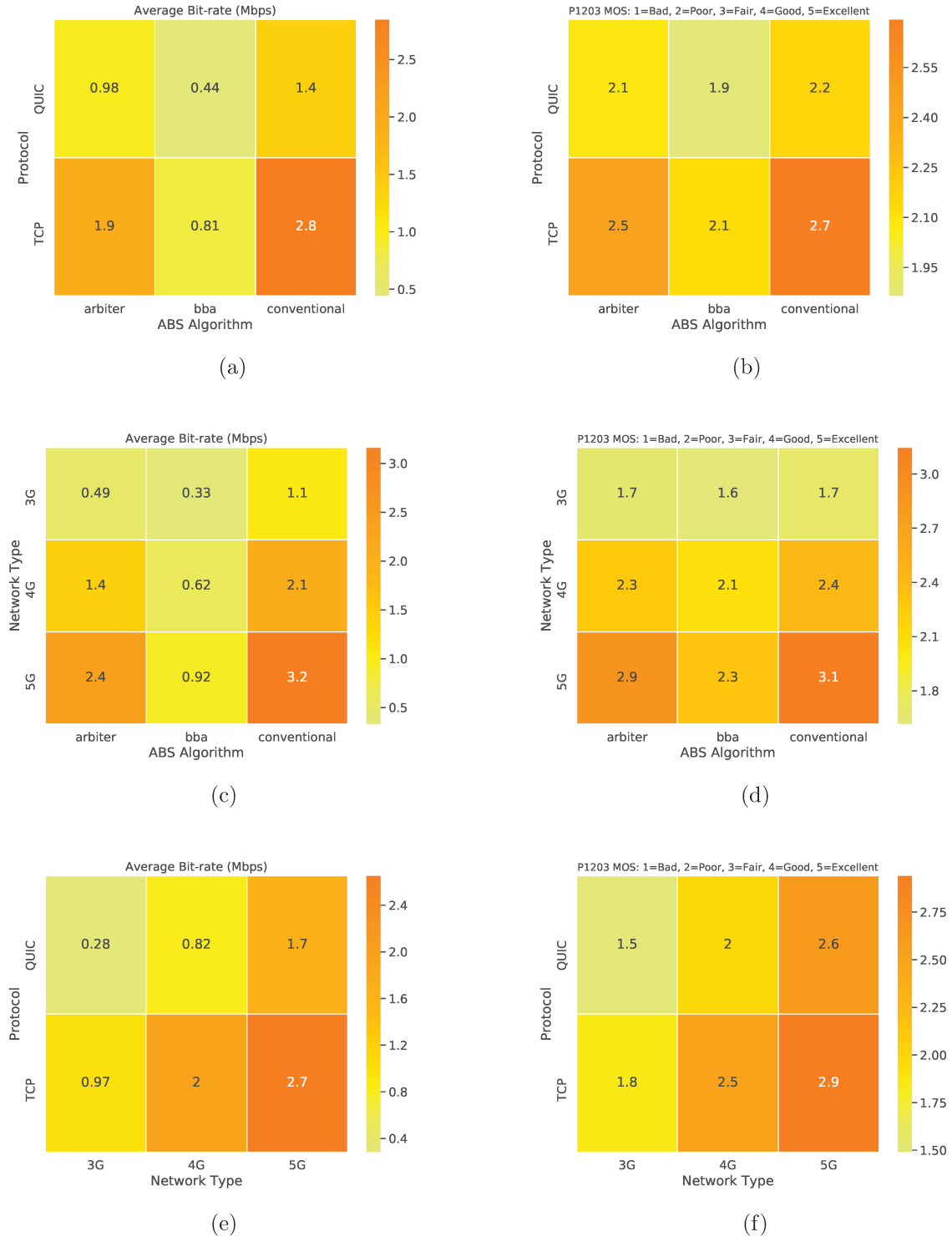


Figure 4.15: Experiment 1: Parallel Client- Average Bit-rate, P1203 MOS: (a) (b) Protocol vs ABS Algorithm, (c) (d) Network Type vs ABS Algorithm and (e) (f) Protocol vs Network Type

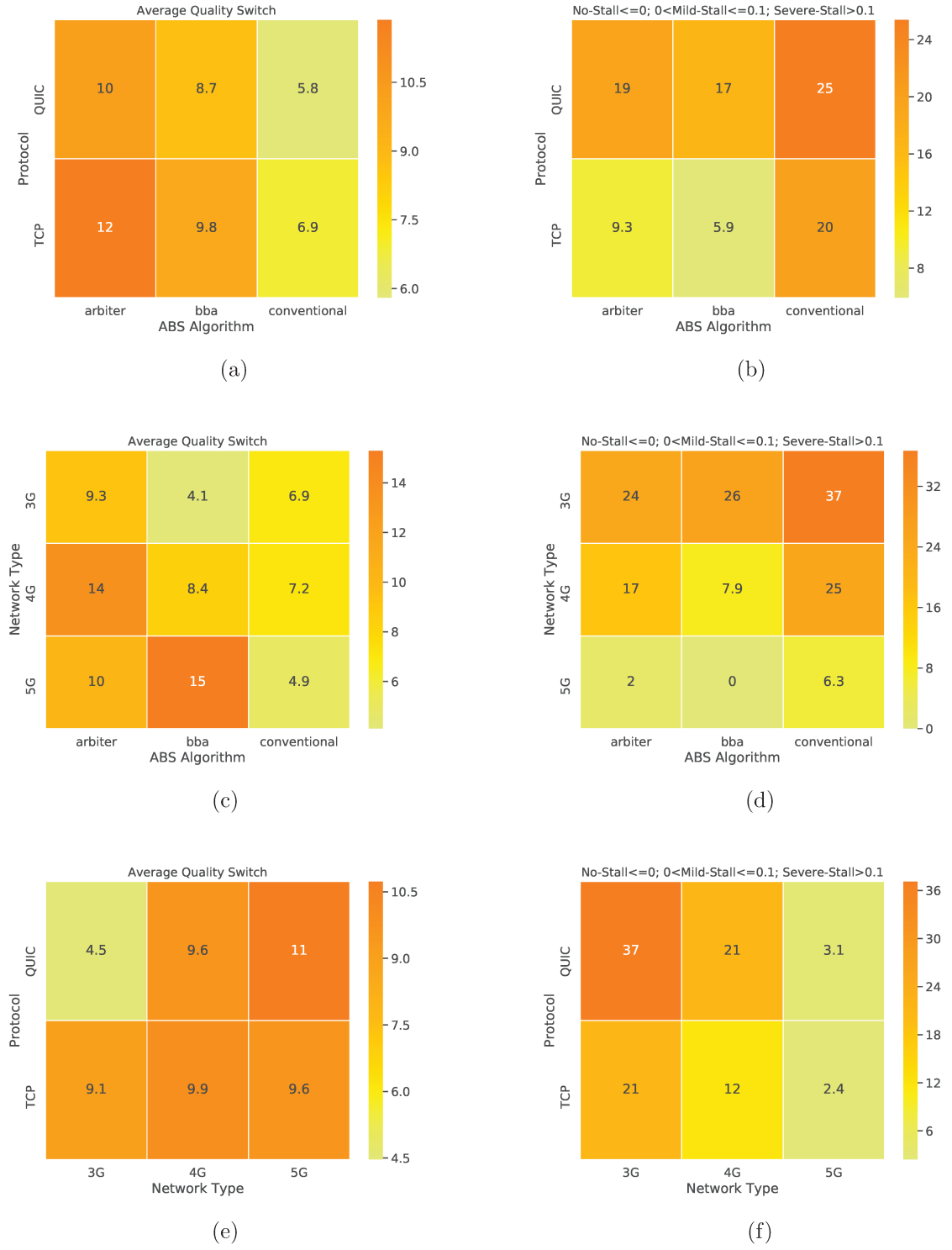
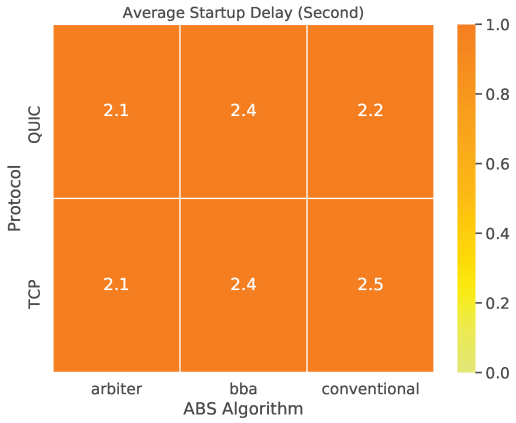
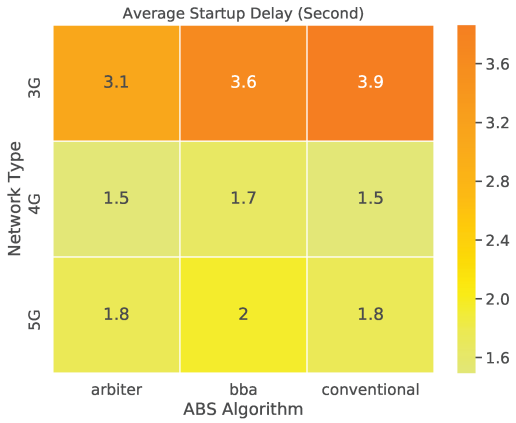


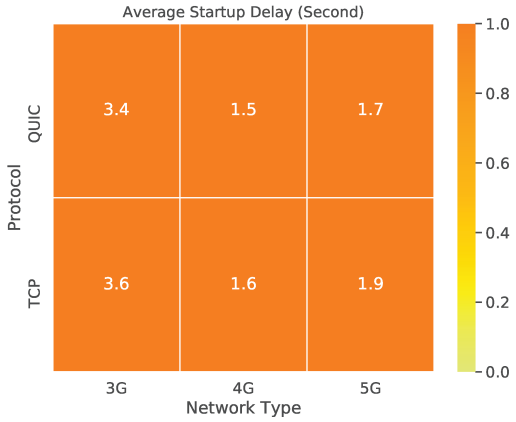
Figure 4.16: Experiment 1: Parallel Client- Quality Switches, Stall Ratio: (a) (b) Protocol vs ABS Algorithm, (c) (d) Network Type vs ABS Algorithm and (e) (f) Protocol vs Network Type



(a)



(b)



(c)

Figure 4.17: Experiment 1: Parallel Client- Startup Delay: (a) Protocol vs ABS Algorithm, (b) Network Type vs ABS Algorithm and (c) Protocol vs Network Type

4.2.2 Experiment 2- QoE Performance over an Unstable Network with Extreme Bandwidth Fluctuation

As stated in [90], the collected 5G trace observed many bandwidth fluctuations due to a lack of 5G base stations across all driving routes which forced the devices to use 4G and 3G. Thus, this experiment was conducted under a single driving mobility trace (Bandwidth: mean=2.56 and std=2.43 Mbps) of the 5G network with 1-second granularity. Linux TC was applied to change the link bandwidth after every 1 second to emulate a restless connection switch scenario. This experiment covers a single DASH client w/ and w/o background (cross) traffic scenario to stream DASH content.

Figure 4.18 and 4.19 describe the performance of single client video streaming in terms of QoE metrics using a scenario of frequent fluctuations in link bandwidth usage. It is noticeable that differences in the output of QoE metrics w/ and w/o background traffic are negligible. To download the desirable bit-rate, each of the ABS algorithms obtained a high bit-rate over TCP, and as a consequence, TCP had higher P1203 MOS values than QUIC. BBA suffered most quality changes in the quality switches scenario, and Conventional suffered minor quality changes over TCP and QUIC.

It is visible in Figure 4.20, which has drawn from one of the random samples of this experiment that Conventional was more constant to select the quality of the segments using TCP, and BBA often changed the segment quality using both TCP and QUIC. Also, Conventional and Arbiter suffered nearly zero stall events over only TCP. Both algorithms rely on bandwidth estimation to select the segments' quality; hence TCP transport facilitates adjusting to the get properly requested segments and filling the player buffer in time. On the other hand, BBA relies on buffer status to decide each segment and holds low quality. Thus, players' buffer always remains fill-up with low-quality segments, leading to no stall event over both TCP and QUIC.

Lastly, the startup delay result is distinct from the previous experiment. In this case, each ABS algorithm took startup playback delays slightly different from each other. The fluctuation of the link bandwidth in the granularity of 1 second might lead to this. In the end, BBA as an ABS algorithm and QUIC as a transport protocol occupied higher startup delay than others. The summary of experiments 1 and 2 concerning the QoE metrics (KPIs) for the best ABS algorithm and transport option is given in the Table 4.9.

Table 4.9: Best ABS Algorithm and Transport Based on QoE Metrics (KPIs) Results

QoE Metrics	**Exp. 1- Single Client**		**Exp. 1- Parallel Client**		**Exp. 2- Single Client**	
	ABS Algorithm	Transport	ABS Algorithm	Transport	ABS Algorithm	Transport
Bit-rate (High)	Conventional	TCP	Conventional	TCP	Conventional	TCP
P1203 MOS (High)	Conventional	TCP	Conventional	TCP	Conventional	TCP
Quality Switch (Stable)	Conventional	-	Conventional	-	Conventional	TCP
Stall Ratio (Low)	BBA	TCP	BBA	TCP	BBA	TCP
Startup Delay (Low)	-	-	-	-	Arbiter	TCP

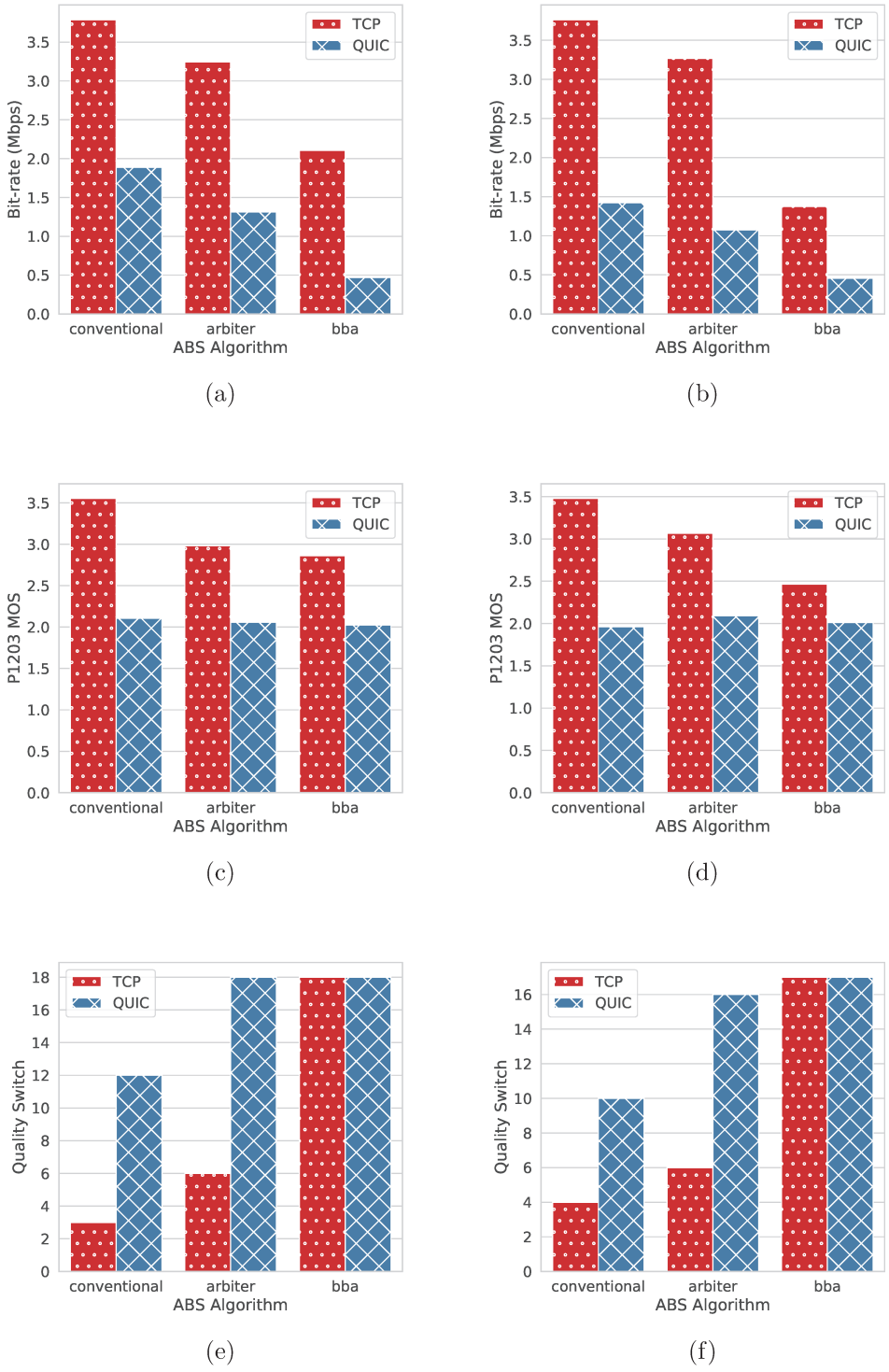


Figure 4.18: Experiment 2: Single Client- Average Bit-rate, P1203 MOS, Quality Switches, Stall Ratio, Startup Delay: (a) (c) (e) w/ Background Traffic, (b) (d) (f) w/o Background Traffic

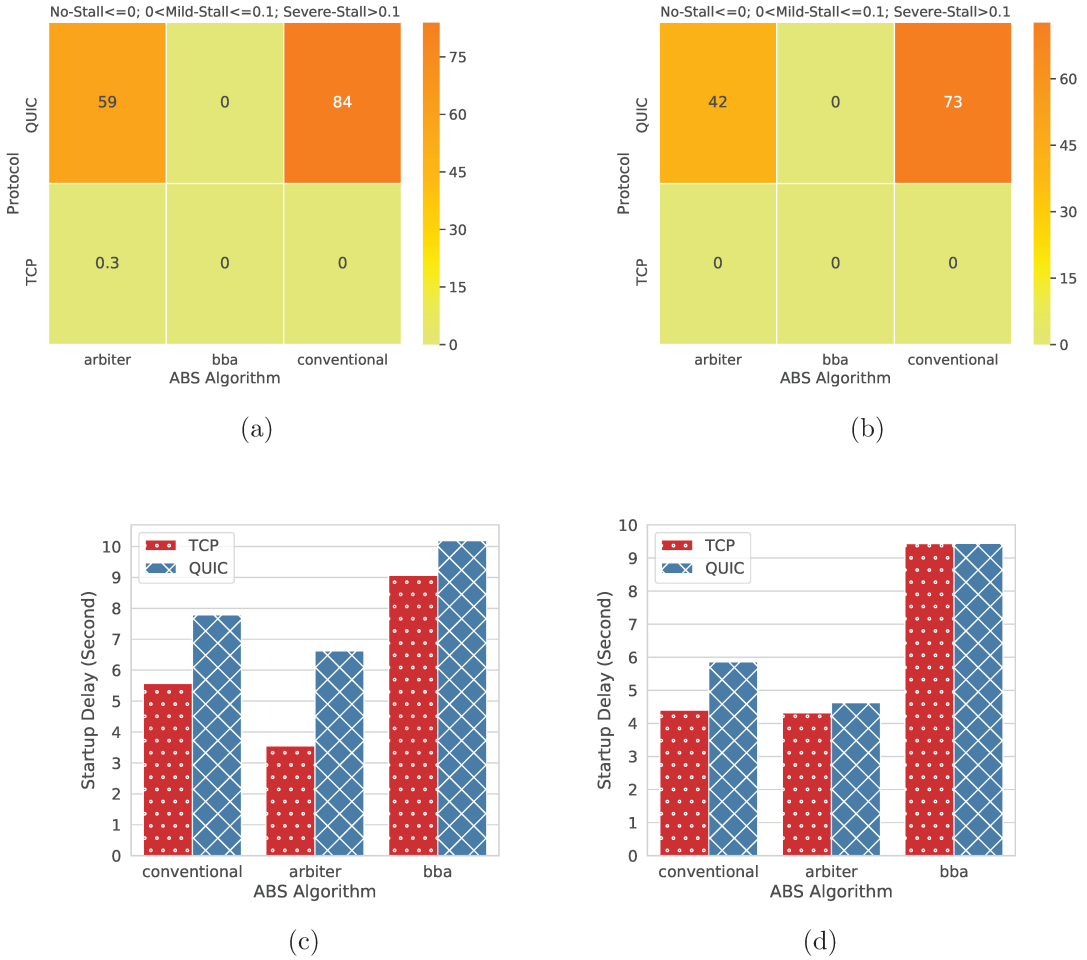
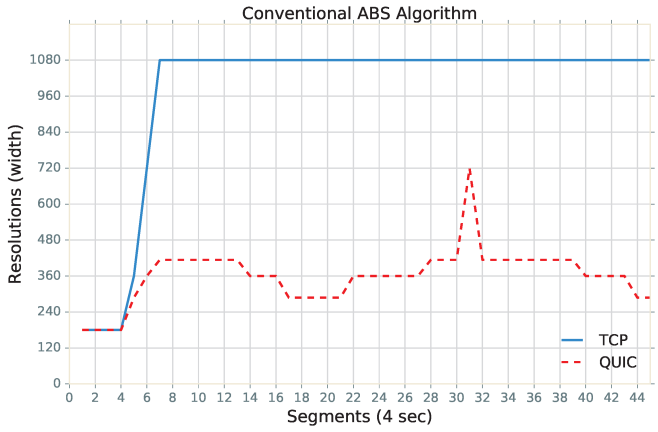
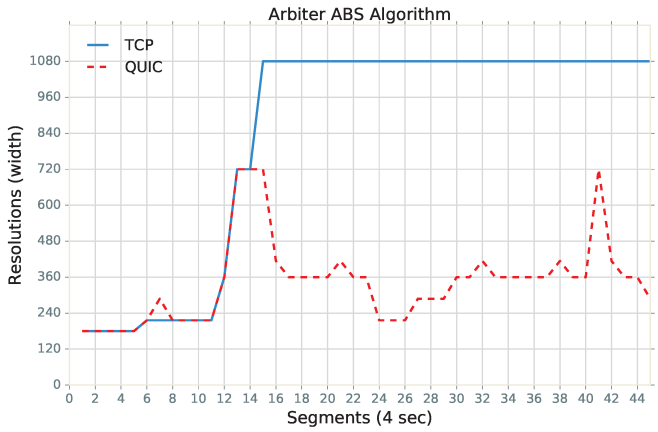


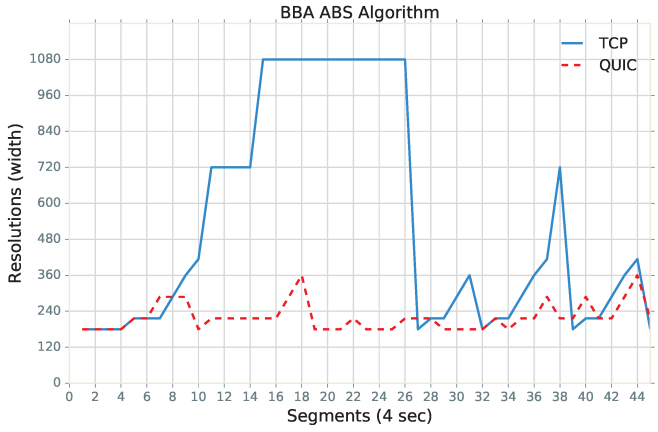
Figure 4.19: Experiment 2: Single Client- Stall Ratio, Startup Delay: (a) (c) w/ Background Traffic, (b) (d) w/o Background Traffic



(a)



(b)



(c)

Figure 4.20: Single Client- Segment Selection of ABS Algorithms in term of Quality (Resolution) During the 3-minute (4 Seconds * 45 Segments) Video Session

Chapter 5

Conclusions, Limitations and Future Work

5.1 Conclusions

This thesis presented a passive network traffic probing technique that allows network operators to assess the client-side obtained experience for DASH-specific video services at the edge computing facility. Also, this thesis provided an empirical video streaming performance study over the different transport protocols. In the following, we concisely summarize the contributions to reflect this thesis's goal and objectives.

5.1.1 Design and Evaluation of Predictive Edge QoE Probe

We designed and evaluated the non-invasive predictive `Edge QoE Probe`'s performance by passively monitoring encrypted network traffic from the network's edge. Our approach played a role in predicting the target end-user perceived QoE for DASH video without requiring end-user direct cooperation by the assistance of the QoS-to-QoE supervised ML correlation model. We stated all the reasonable steps to identify the video traffic and sessions and conducted an extensive experiment using a fully reproducible emulation-based testbed. Numerous cellular network traces information (e.g., downlink bandwidth) was used to realistically emulate mobility scenarios on the controlled testbed for emulation purposes. We also played DASH video utilizing different client-side state-of-the-art ABS algorithms over TCP (HTTPs) and QUIC transport that facilitate end-to-end encryption.

The proposed design demonstrated a lightweight, fine-grained temporal (three-time window-based, i.e., *current*, *trend* and *session*) network-level QoS feature extraction by observing bi-directional IP packet header information from the edge node and capable of predicting per 0.5-second interval video streaming QoE metrics without DPI and video segment identification. For real-time video QoE KPIs estimation, we emphasized more on displayed video quality. Thus, we considered only resolution and bit-rate KPIs as other KPIs rarely observed (e.g., stall events) or hardly measurable (e.g., quality switches).

As far as we are aware, our approach by now the shortest granularity for in-network QoE inferring of encrypted video streaming. Such real-time estimation strongly impacts the network management, specifically in the self-governing CCL system, to be used as a threshold for taking reactive performance diagnosis and resource allocation actions. Moreover, our proposed method estimated per-session several QoE KPIs, namely average resolution, average bit-rate, quality switches, stall ratio, startup delay, and P1203 MOS, by aggregating real-time QoS features.

Such estimation is highly relevant to network operators to review the SLA for proactive network capacity planning and configuration. Hereby, both real-time and per-session in-network QoE inferring enables proactive and reactive QoE-aware network traffic management.

We built datasets over TCP and QUIC transport separately for a specific DASH video content and benchmarked different supervised ML models. We concluded that Random Forest, a tree-based ensemble learning method, provides the most suitable fine-grained regression and multi-class classification task model. The Random Forest-based model accurately inferred real-time and per-session video QoE KPIs by taking reasonable training time in our work. Furthermore, we explored the relative QoS feature importance using the Random Forest-based model. We found that the top 15 *session* window QoS input features-based model achieved a higher inference accuracy score for real-time QoE estimation. We also showed per-session QoS feature importance on each QoE KPIs estimation.

The study of the QoS input feature's importance shows potential feature influences on specific QoE KPIs. And, it indicates that more carefully crafting the QoS features will lead our proposed `Edge QoE Probe` to become more efficient in processing time and memory consumption.

5.1.2 Evaluation of QoE Study over TCP and QUIC Transport

QUIC, a relatively new protocol with the promise of performance improvement over the widely used TCP, motivated us to reconsider an ideal transport protocol for adaptive video streaming service. Thus, we investigated the three strategies (Rate, Buffer, and Hybrid) based ABS algorithm streaming performance over TCP and QUIC using different cellular network traces information (e.g., downlink bandwidth) in a controlled testbed alongside the QoE estimation.

Our preliminary study showed all ABS algorithms using TCP transport achieved a high quality of video streaming performance (e.g., specifically high average bit-rate and P1203 MOS, and low stall ratio) under varying network conditions (e.g., stable and unstable) and verified the earlier work [76] conclusion. We also found the rate-based Conventional ABS algorithm provides considerably better performance compared to other algorithms.

The poor performance over QUIC transport indicates that the traditional state-of-the-art ABS algorithms were built mainly on TCP in mind. As a result, despite HOL issues and handshake latency in HTTP/1.1, TCP still performs better than HTTP/3 over QUIC for adaptive video streaming.

To deal with the IETF[1] mentioned QUIC features, state-of-the-art ABS algorithms require modification in terms of video segment requests to embrace the potential benefit of QUIC multiplexing, disable HOL blocking, congestion control, and service migration.

5.2 Limitations

While conducting QoE estimation using the supervised ML approach for the generated datasets by emulation-based testbed, we considered few limitations. Besides, this thesis noted the limitation for the QoE performance study. The overall enlisted limitations, as given in below, provide opportunities for improvement in further work.

- **Lack of Generalization**: The dataset used in this thesis was obtained using a headless goDASH player based on a high-fidelity emulated controlled testbed. This work more

[1] https://datatracker.ietf.org/doc/draft-ietf-quic-recovery/

emphasized target end-user QoE and specific video service rather than generalization. In reality, QoS's impact on QoE may vary as end-users can stream video from different OTT platforms (e.g., Netflix, YouTube, Amazon) via various devices (e.g., Mobile, PC, Tablet). Besides, in devices, the scenarios are more complicated because of the diversity of browsers (e.g., Chrome, Mozilla), streaming apps, mobile platforms (e.g., Android, IOS). Therefore the used QoE prediction model in this thesis is unable to manage such diversity.

- **Model Re-evaluation**: In this thesis, supervised ML model trained based on label dataset where ground truth obtained from test device (e.g., goDASH player running over mininet-wifi station) for predefined network conditions and adaptive streaming logic (e.g., ABS algorithms). Over time, in reality, it is expected that video adaptation schemes by content providers and network conditions can change, which would limit the comprehensive applicability of the trained model for future usage. Therefore, the model requires periodically re-evaluating over a while to assess the model's performance with new ground truth. The challenges related to this aspect are to define a specific threshold (e.g., level of accuracy) to re-evaluate the model's effectiveness and amount of data need with new ground truth for retraining the model if the existing model fails to obtain the particular threshold.

- **MOS Calculation**: In this thesis, the output of MOS was based on ITU-T Rec. P.1203.1 standardization considering device type- PC, display size- 1920x1080, and viewing distance- 150cm. Based on a different device type, screen size, and viewing distance, MOS output will be changed, which would affect the model's performance.

- **Lack of Root Cause Localization**: To take proper reactive action, the network operators need to identify the potential root cause (e.g., congestion in specific backhaul links), which causes QoE degradation. Our approach was more focused on inferring QoE rather than detecting the reason for QoE impairments. However, network operators can employ a decision tree model that allows identifying root causes with little effort.

- **Lack of Integration with Radio Access Information**: This work was leveraged by the concept of edge computing, which considers a combination of both backhaul and fronthual networks. However, our approach lacks fronthual information (e.g., RAN analytics). Hence, the current system limits tracking the issues that lead to low QoE in the wireless network link.

- **Practical Deployment**: Our designed QoS features technique and ML model require evaluation for practical deployment. This thesis lacks assess the computational complexity for run-time operation, such as time requires to update the features set for per-packet/time interval during feature extraction, QoE metrics estimation time for per time interval by the trained model. Such run-time evaluation will help to justify our proposed real-time video streaming QoE metrics estimation for the 0.5-second interval in the case of the practical consideration. Besides, in virtualization, how much resources (e.g., CPU, memory, storage) would require such `Edge QoE Probe` as a VNF remains vague and needs more studies.

- **Lack of Indicating the Variability of QoE Study**: For the QoE study over TCP and QUIC transport, this thesis repeated the experiment three-times and considered only the average values, which currently lacks to present the overall variability (e.g., confidence intervals) of the results (e.g., QoE metrics).

5.3 Future Work

To make feasible the `Edge QoE Probe` for run-time operation from network-level temporal QoS feature extraction as well as novel QoS feature extraction technique and obtain more meaningful empirical QoE performance study over TCP and QUIC transport, the scopes of future work are given below.

5.3.1 Integrating Edge QoE Probe to the CCL Platform

As future work, we plan to integrate our `Edge QoE Probe`, specifically network-level QoS feature extraction approach, to the intent-based CCL platform [108]. A high-level overview of the CCL architectural design is given in Figure 5.1. It contains the following components.

1. **Collector**: This component executes a non-invasive network-level measurement method based on traffic mirroring processed by a passive probe running at the edge node (e.g., MEC).

2. **QoE Estimator**: The QoE Estimator component (e.g., supervised ML-based) performs inferences of video QoE metrics/KPIs from network-level QoS input features received from the Collector component.

3. **Policy-driven Orchestrator**: This component stands on adaptive policies in a control loop approach that automatically manages a set of orchestrated actions (e.g., reactive) to assure the end-to-end network service quality.

4. **Actuator**: This component is responsible for closing the smart control loop by translating the orchestrator's high-level actions to low-level actions (e.g., network device commands of the underlying infrastructure).

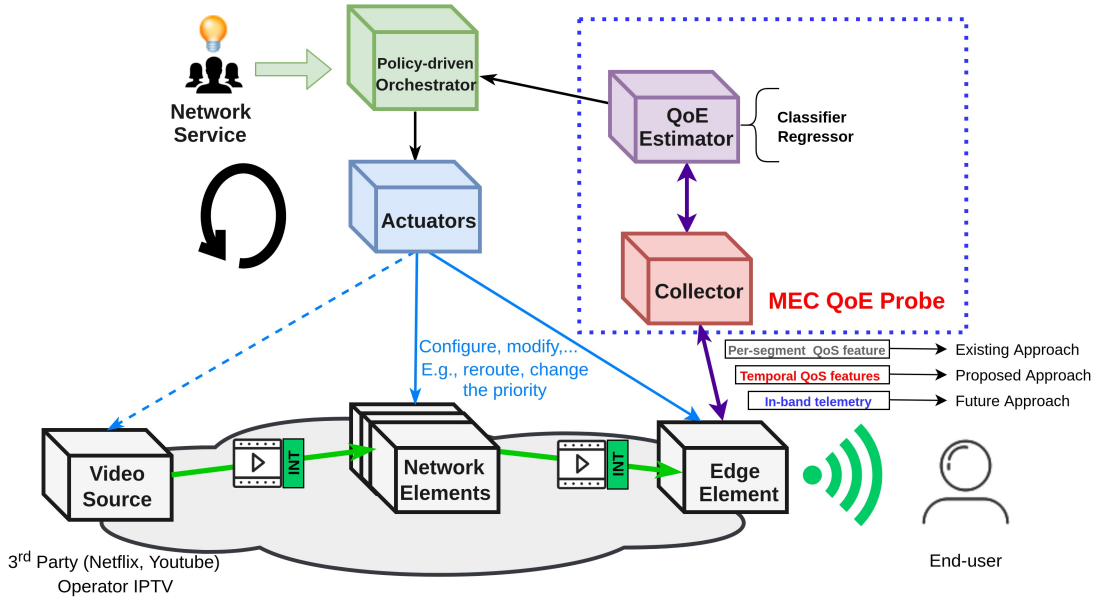


Figure 5.1: CCL Architectural Overview

Currently, the Collector component relies on video segment detection from unencrypted traffic (e.g., HTTP/1.1 over TCP) to extract (e.g., python-based script) per video segments QoS features (e.g., uplink RTT, downlink throughput, and packets). The segment detection and per-segments QoS features require more processing time and unfeasible when multiple segments are downloaded parallel (e.g., HTTP/2 or HTTP/3 multiplexing). Therefore, we focus on integrating our lightweight temporal QoS feature extraction technique to make the Collector component more robust and fine-grained (e.g., short time interval).

5.3.2 In-band Network Telemetry (INT) as Potential Future Source of Network QoS KPIs

Another novel idea for the future is the potential utilization of more fine-grained QoS information to estimate QoE, using INT (e.g., using the P4 language) [109] [110]. It has capabilities to passively monitor network and traffic information (including the status of a network device-hop latency, queue occupancy) from the data plane beyond the programmability of the SDN control plane. The core idea of INT (See Figure 5.2) is to write the network status (from INT source) into the header of a data packet to guarantee the granularity of monitoring the network at the packet level. The telemetry data will be attached to the network packet. When the network packet comes to the INT sink point, the load data will send to the user, and the telemetry data will send to the network monitoring probe. INT makes it easier to gather and analyze information such as when a packet enters and leaves the network, the packet arrival rate at a specific hop, what path a packet takes, how long the packet spends on each hop, and which switches experience congestion. The edge node (e.g., MEC) would act as a sink to send the telemetry data to the QoE probe to estimate objective QoE KPIs by ML approach and take further action to optimize the network. At present, there is a basic tutorial available on INT at the P4-Github[2] repository. It would be interesting to extend that tutorial as a future task by adding more metadata to make it feasible for QoE inferring.

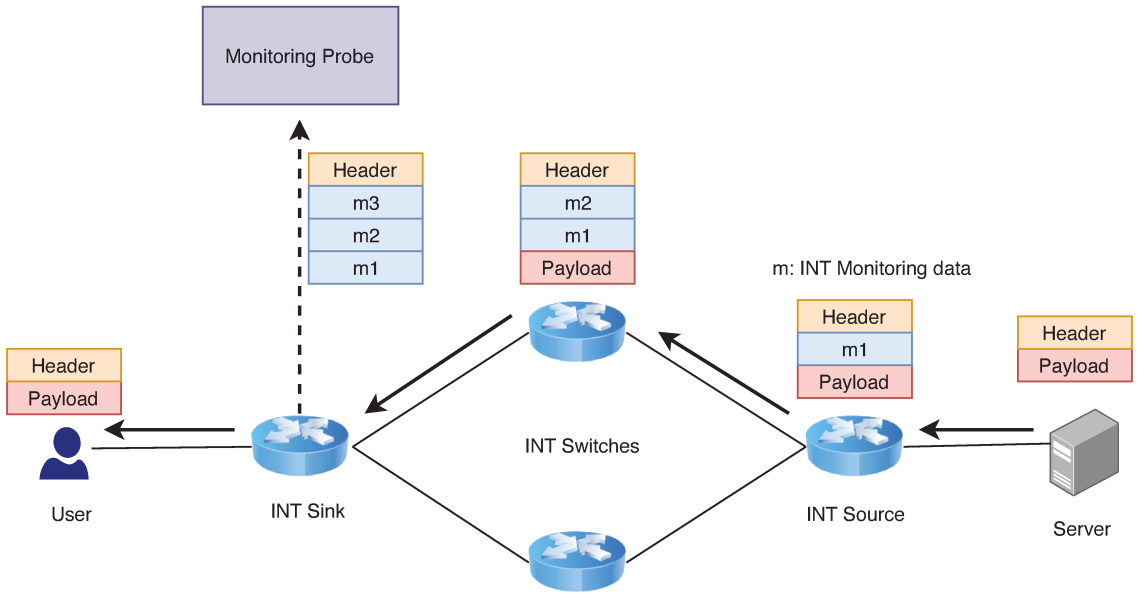


Figure 5.2: In-band Network Telemetry (INT) Overview

[2]https://github.com/p4lang/tutorials/tree/master/exercises/mri

5.3.3 Extensive Adaptive Streaming QoE Study over TCP and QUIC

In the future, we plan to extend our empirical QoE study by adding more DASH segment size content, different buffer levels of DASH player, more realistic lousy network condition, and the alternative implementation of QUIC transport taking into account the latest efforts made by IETF[3]. We will increase the number of experiments repetition for comparing the adaptive streaming QoE using HTTP/2 over TCP and HTTP/3 over QUIC (including different open-source implementations of QUIC transport, i.e., aioquic[4], lsquic[5], picoquic[6], mvfst[7], quiche[8], etc.). Moreover, it would be interesting to find out any novel approaches alongside the existing findings to modify ABS algorithms' segment choice decision to achieve better performance over QUIC transport.

[3]https://datatracker.ietf.org/doc/draft-ietf-quic-transport/
[4]https://github.com/aiortc/aioquic
[5]https://github.com/litespeedtech/lsquic
[6]https://github.com/private-octopus/picoquic
[7]https://github.com/facebookincubator/mvfst
[8]https://github.com/cloudflare/quiche

Bibliography

[1] N. S. Cisco, "Cisco annual internet report (2018–2023)," *White Paper*, 2020.

[2] Sandvine, "The mobile internet phenomena report," *Technical Report*, 2020.

[3] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017-2022," *White Paper*, 2018.

[4] C. V. Networking, "Cisco visual networking index: Forecast and methodology, 2016–2021," *White Paper*, 2017.

[5] P. Jonsson, S. Carson, G. Blennerud, J. Kyohun Shim, B. Arendse, A. Husseini *et al.*, "Ericsson mobility report. 2019," *Ericsson: Stockholm, Sweden*, 2019.

[6] Conviva, *Conviva's State of Streaming Q1 2020*, 2020, https://www.conviva.com/state-of-streaming.

[7] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for http live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, 2014.

[8] 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," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 526–565, 2020.

[9] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for qoe management of multimedia services," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2s, p. 29, 2018.

[10] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[11] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, 09 2015.

[12] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks," in *2015 European Conference on Networks and Communications (EuCNC)*, 2015, pp. 239–243.

[13] P. Juluri, L. Plissonneau, and D. Medhi, "Pytomo: a tool for analyzing playback quality of youtube videos," in *Proceedings of the 23rd International Teletraffic Congress*. International Teletraffic Congress, 2011, pp. 304–305.

[14] R. Schatz, T. Hoßfeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2012, pp. 358–364.

[15] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "Buffest: Predicting buffer conditions and real-time requirements of http (s) adaptive streaming clients," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 76–87.

[16] M. Bishop *et al.*, "Hypertext transfer protocol version 3 (http/3)," *Internet Engineering Task Force, Internet-Draft draft-ietf-quic-http-27*, 2020.

[17] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Vicrypt: Real-time, fine-grained prediction of video quality from encrypted streaming traffic," in *ACM Internet Measurement Conference (IMC) 2019*, 2019.

[18] T. Stockhammer *et al.*, "Dynamic adaptive streaming over http-design principles and standards," in *Proceedings of the second annual ACM conference on Multimedia systems*, vol. 2014. New York, USA: ACM, 2011, pp. 2–4.

[19] R. Fielding and J. Reschke, "Rfc 7230: Hypertext transfer protocol (http/1.1): Message syntax and routing," *Internet Engineering Task Force (IETF)*, vol. 6, no. 01, 2014.

[20] R. Marx, T. De Decker, P. Quax, and W. Lamotte, "Of the utmost importance: Resource prioritization in http/3 over quic." in *WEBIST*, 2019, pp. 130–143.

[21] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[22] M. ETSI, "Mobile edge computing-introductory technical white paper," *etsi2014mobile, no. Issue*, 2014.

[23] M.-A. E. Computing, "Phase 2: Use cases and requirements," *Standard ETSI GS MEC*, vol. 2, p. V2, 2018.

[24] S. M. Patrick Le Callet and e. Andrew Perkis, "Qualinet white paper on definitions of quality of experience(2012)," *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, Lausanne, Switzerland, Version 1.2, March 2013.

[25] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 362–373.

[26] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[27] L. Skorin-Kapov, M. Varela, T. Hossfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for qoe management of multimedia services," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, pp. 1–29, 05 2018.

[28] A. Takahashi, D. Hands, and V. Barriac, "Standardization activities in the itu for a qoe assessment of iptv," *IEEE Communications Magazine*, vol. 46, no. 2, pp. 78–84, 2008.

[29] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, "Crowdsourcing multimedia qoe evaluation: A trusted framework," *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1121–1137, 2013.

[30] Y. Chen, K. Wu, and Q. Zhang, "From qos to qoe: A tutorial on video quality assessment," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1126–1165, Secondquarter 2015.

[31] A. Raake, J. Gustafsson, S. Argyropoulos, M.-N. Garcia, D. Lindegren, G. Heikkilä, M. Pettersson, P. List, and B. Feiten, "Ip-based mobile and fixed network audiovisual media services," *IEEE Signal Processing Magazine*, vol. 28, pp. 68–79, 11 2011.

[32] 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," *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22 243–22 266, 2017.

[33] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*. ACM, 2016, pp. 513–526.

[34] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Flowcover: Low-cost flow monitoring scheme in software defined networks," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 1956–1961.

[35] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1842–1866, 2017.

[36] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[37] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "Elastic: A client-side controller for dynamic adaptive streaming over http (dash)," in *2013 20th International Packet Video Workshop*, 2013, pp. 1–8.

[38] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (http/2)," 2015.

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

[40] A. Neal, B. Naughton, C. Chan, N. Sprecher, and S. Abeta, "Mobile edge computing (mec); technical requirements," *ETSI, Sophia Antipolis, France, White Paper no. DGS/MEC-002*, 2016.

[41] M. Seufert, N. Wehner, F. Wamser, P. Casas, A. D'Alconzo, and P. Tran-Gia, "Unsupervised qoe field study for mobile youtube video streaming with yomoapp," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[42] H. Nam, K.-H. Kim, D. Calin, and H. Schulzrinne, "Youslow: a performance analysis tool for adaptive bitrate video streaming," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 111–112.

[43] D. Joumblatt, J. Chandrashekar, B. Kveton, N. Taft, and R. Teixeira, "Predicting user dissatisfaction with internet application performance at end-hosts," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 235–239.

[44] K.-T. Chen, C.-C. Tu, and W.-C. Xiao, "Oneclick: A framework for measuring network quality of experience," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 702–710.

[45] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Mimic: Using passive network measurements to estimate http-based adaptive video qoe metrics," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 2017, pp. 1–6.

[46] T. Mangla, E. Zegura, M. Ammar, E. Halepovic, K.-W. Hwang, R. Jana, and M. Platania, "Videonoc: assessing video qoe for network operators using passive measurements," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018, pp. 101–112.

[47] A. Farshad, P. Georgopoulos, M. Broadbent, M. Mu, and N. Race, "Leveraging sdn to provide an in-network qoe measurement framework," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2015, pp. 239–244.

[48] R. Huysegems, B. De Vleeschauwer, K. De Schepper, C. Hawinkel, T. Wu, K. Laevens, and W. Van Leekwijck, "Session reconstruction for http adaptive streaming: Laying the foundation for network-based qoe monitoring," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IEEE Press, 2012, p. 15.

[49] T. Wu, R. Huysegems, and T. Bostoen, "Scalable network-based video-freeze detection for http adaptive streaming," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*. IEEE, 2015, pp. 95–104.

[50] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Using session modeling to estimate http-based video qoe metrics from encrypted network traffic," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1086–1099, 2019.

[51] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "Youtube qoe estimation based on the analysis of encrypted network traffic using machine learning," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[52] 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 *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 233–239.

[53] V. Vasilev, J. Leguay, S. Paris, L. Maggi, and M. Debbah, "Predicting qoe factors with machine learning," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[54] M. J. Khokhar, T. Ehlinger, and C. Barakat, "From network traffic measurements to qoe for internet video," in *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 2019, pp. 1–9.

[55] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1331–1339.

[56] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-time qoe detection for encrypted youtube traffic," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 48–59.

[57] P. Schmitt, F. Bronzino, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *arXiv preprint arXiv:1901.05800*, 2019.

[58] 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 *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2019, pp. 76–81.

[59] 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 *Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019, pp. 1–6.

[60] I. Orsolic and L. Skorin-Kapov, "A framework for in-network qoe monitoring of encrypted video streaming," *IEEE Access*, vol. 8, pp. 74 691–74 706, 2020.

[61] S. Tang, X. Qin, and G. Wei, "Analysis on the state of mobile http video streaming at the client-side," in *2015 International Conference on Wireless Communications & Signal Processing (WCSP)*. IEEE, 2015, pp. 1–6.

[62] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support," *Sensors*, vol. 18, no. 2, p. 329, 2018.

[63] L. Dinh-Xuan, M. Seufert, F. Wamser, and P. Tran-Gia, "Study on the accuracy of qoe monitoring for http adaptive video streaming using vnf," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 999–1004.

[64] X. Xu, J. Liu, and X. Tao, "Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation," *IEEE Access*, vol. 5, pp. 16 406–16 415, 2017.

[65] C. Ge, N. Wang, S. Skillman, G. Foster, and Y. Cao, "Qoe-driven dash video caching and adaptation at 5g mobile edge," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM, 2016, pp. 237–242.

[66] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, 2017.

[67] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Enhancing qoe-aware wireless edge caching with software-defined wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6912–6925, 2017.

[68] R. Behravesh, D. F. Perez-Ramirez, A. Rao, D. Harutyunyan, R. Riggio, and R. Steinert, "Ml-driven dash content pre-fetching in mec-enabled mobile networks," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–7.

[69] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-based architecture for improved adaptive http video delivery," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2016, pp. 1–6.

[70] M. Kim and K. Chung, "Edge computing assisted adaptive streaming scheme for mobile networks," *IEEE Access*, 2020.

[71] C. Ge and N. Wang, "Real-time qoe estimation of dash-based mobile video applications through edge computing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 766–771.

[72] H. Zheng, Y. Zhao, X. Lu, and R. Cao, "A mobile fog computing-assisted dash qoe prediction scheme," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[73] C. Tselios and G. Tsolis, "On qoe-awareness through virtualized probes in 5g networks," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2016, pp. 159–164.

[74] S. Peng, J. O. Fajardo, P. S. Khodashenas, B. Blanco, F. Liberal, C. Ruiz, C. Turyagyenda, M. Wilson, and S. Vadgama, "Qoe-oriented mobile edge service management leveraging sdn and nfv," *Mobile Information Systems*, vol. 2017, 2017.

[75] C. Timmerer and A. Bertoni, "Advanced transport options for the dynamic adaptive streaming over http," *arXiv preprint arXiv:1606.00264*, 2016.

[76] D. Bhat, A. Rizk, and M. Zink, "Not so quic: A performance study of dash over quic," in *Proceedings of the 27th workshop on network and operating systems support for digital audio and video*, 2017, pp. 13–18.

[77] S. Arisu and A. C. Begen, "Quickly starting media streams using quic," in *Proceedings of the 23rd Packet Video Workshop*, 2018, pp. 1–6.

[78] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at quic: an approach for rigorous evaluation of rapidly evolving transport protocols," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 290–303.

[79] T. Zinner, S. Geissler, F. Helmschrott, and V. Burger, "Comparison of the initial delay for video playout start for different http-based transport protocols," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 1027–1030.

[80] B. Li, C. Wang, Y. Xu, and Z. Ma, "An mmt based heterogeneous multimedia system using quic," in *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT)*. IEEE, 2016, pp. 129–133.

[81] B. Hayes, Y. Chang, and G. Riley, "Omnidirectional adaptive bitrate media delivery using mptcp/quic over an sdn architecture," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[82] D. Bhat, R. Deshmukh, and M. Zink, "Improving qoe of abr streaming sessions through quic retransmissions," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1616–1624.

[83] M. Nguyen, H. Amirpour, C. Timmerer, and H. Hellwagner, "Scalable high efficiency video coding based http adaptive streaming over quic," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, 2020, pp. 28–34.

[84] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 384–389.

[85] D. Raca, M. Manifacier, and J. J. Quinlan, "godash-go accelerated has framework for rapid prototyping," 2020.

[86] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-transport-29, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-29

[87] J. J. Quinlan and C. J. Sreenan, "Multi-profile ultra high definition (uhd) avc and hevc 4k dash datasets," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 375–380.

[88] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.

[89] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g lte dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 460–465.

[90] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 303–308.

[91] D. Raca, Y. Sani, C. J. Sreenan, and J. J. Quinlan, "Dashbed: a testbed framework for large scale empirical evaluation of real-time dash in wireless scenarios," in *Proceedings of the 10th ACM Multimedia Systems Conference*. ACM, 2019, pp. 285–290.

[92] 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," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[93] Y. Sani, A. Mauthe, and C. Edwards, "Modelling video rate evolution in adaptive bitrate selection," in *2015 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2015, pp. 89–94.

[94] A. H. Zahran, D. Raca, and C. J. Sreenan, "Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2716–2728, 2018.

[95] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)," in *2013 20th International Packet Video Workshop*. IEEE, 2013, pp. 1–8.

[96] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and validating user experience model for dash video streaming," *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 651–665, 2015.

[97] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "Qoe-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 2, p. 28, 2016.

[98] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M.-N. Garcia *et al.*, "Http adaptive streaming qoe estimation with itu-t rec. p. 1203: open databases and software," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 466–471.

[99] 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 *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[100] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "Qoe-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 2, pp. 1–24, 2015.

[101] Z. Duanmu, A. Rehman, and Z. Wang, "A quality-of-experience database for adaptive video streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 474–487, 2018.

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

[103] L. Yu, T. Tillo, and J. Xiao, "Qoe-driven dynamic adaptive video streaming strategy with future information," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 523–534, 2017.

[104] F. Bronzino, P. Schmitt, S. Ayoubi, N. Feamster, R. Teixeira, S. Wasserman, and S. Sundaresan, "Lightweight, general inference of streaming video quality from encrypted traffic," *arXiv preprint arXiv:1901.05800*, 2019.

[105] W. Song, D. Tjondronegoro, and M. Docherty, "Saving bitrate vs. pleasing users: where is the break-even point in mobile video quality?" in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 403–412.

[106] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a qoe model for http adaptive streaming," in *2014 sixth international workshop on quality of multimedia experience (qomex)*. IEEE, 2014, pp. 111–116.

[107] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

[108] C. E. Rothenberg, D. A. Lachos Perez, N. F. Saraiva de Sousa, R. V. Rosa, R. U. Mustafa, M. T. Islam, and P. H. Gomes, "Intent-based control loop for dash video service assurance using ml-based edge qoe estimation," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 353–355.

[109] C. Kim, A. Sivaraman, N. P. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," 2015.

[110] F. E. Bustamante, D. Clark, and N. Feamster, "Workshop on tracking quality of experience in the internet: Summary and outcomes," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 1, pp. 55–60, 2017.

Appendix A

Publications

- C. E. Rothenberg, D. A. Lachos Perez, N. F. Saraiva de Sousa, R. V. Rosa, R. U. Mustafa, **M. T. Islam**, and P. H. Gomes, "Intent-based control loop for dash video service assurance using ml-based edge qoe estimation," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 353–355.

- R. U. Mustafa, **M. T. Islam**, C. Rothenberg, S. Ferlin, D. Raca, and J. J. Quinlan, "Dash qoe performance evaluation framework with 5g datasets," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–6.

- **M. Islam** and C. Rothenberg, "Has based empirical qoe study over tcp and quic on diverse networks," in *Anais do VII Workshop Pre-IETF*. SBC, 2020, pp. 29–42.

- N. F. Saraiva de Sousa, **M. T. Islam**, R. U. Mustafa, D. A. Lachos Perez, C. E. Rothenberg, and P. H. Gomes, "Machine learning-assisted closed-control loops for automated healing in multi-domain zero-touch networks," *Manuscript submitted for publication.*