Quality of Experience for Dynamic Adaptive Streaming over HTTP in Mobile Devices: Assessment, Modeling and Improvement

Hema Kumar Yarnagula

Quality of Experience for Dynamic Adaptive Streaming over HTTP in Mobile Devices: Assessment, Modeling and Improvement

Thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Hema Kumar Yarnagula

Under the supervision of

Dr. Venkatesh Tamarapalli

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

February 2022

This thesis is dedicated to the memory of my Father

- whose blessings, inspriation, unconditional love and support made my path of sucess.

Declaration

I hereby certify that

a. The work contained in this thesis is original and has been done by myself and the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.

d. No part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.

Date : ____/_____/_____ **Hema Kumar Yarnagula**

Place: Guwahati, India

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
गुवाहाटी - 781039
Indian Institute of Technology Guwahati
Guwahati - 781039

Dr. Venkatesh Tamarapalli
Associate Professor

: +91 361 258 2366
: +91 361 269 2787
: t.venkat@iitg.ac.in

Department of Computer Science and Engineering

Thesis Certificate

This is to certify that the thesis entitled **"Quality of Experience for Dynamic Adaptive Streaming over HTTP in Mobile Devices: Assessment, Modeling and Improvement"** being submitted by **Hema Kumar Yarnagula** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bonafide research work carried out by him under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date : ____/____/_____

Place: Guwahati, India

Dr. Venkatesh Tamarapalli
(Thesis Supervisor)

Acknowledgments

"No one who achieves success does so without acknowledging the help of others. The wise and confident acknowledge this help with gratitude."

This dissertation would not have seen the light of day had there not been the encouragement, generous assistance, and guidance from an array of helpful individuals. I am grateful to every special person in my life who had offered timely help to me when things appeared to be impossible and contributed in some way to the process leading to my doctoral dissertation.

I would like to express my deepest gratitude, first and foremost, to my doctoral advisor Dr. Venkatesh Tamarapalli for his encouragement, endless support, advice and words of wisdom throughout my doctoral study. I am deeply thankful for the opportunity he gave me to explore my research interests, as well as for his timely direction and insights that kept my pace steady. More than that, throughout the journey, his expertise in system building and insightful feedback have unblocked many technical obstacles of my research study as I struggled through learning curves. His unique perspectives, not just from the technical but especially from the personal point of view, have enlightened the blindspots of my thinking process and inspired me to explore outside the box to overcome my weaknesses as numerous life-changing events unfolded in my personal life. Words will fall short to fully express my gratitude for the genuine compassion and understanding that he had shown during the last several years and for giving me the room to process these changes and evolve as a better person. I feel very privileged to have had the opportunity to learn from and work with him since our first greeting on August 2nd, 2014.

It is not easy to overstate my gratitude to Prof. Deep Medhi for his support, invaluable feedback, his great efforts to explain things clearly and simply. I would like to thank him for all of the advice he has passed along during the manuscript writing process. I am fortunate enough to have him as a mentor throughout my doctoral study.

I would like to take this opportunity to place on record my sincere thanks to the thesis examiners who gave their valuable suggestions for its improvement. I would also like to take this opportunity to thank my doctoral committee members, Prof. Diganta Goswami, Dr. Samit Bhattacharya, Dr. Arnab Sarkar and Dr. John Jose, for their time, insightful comments and valuable suggestions on my research work. I thank them for being very supportive of my research and for their constructive critiques that have contributed significantly to my dissertation.

I am thankful to my wonderful collaborators and co-authors, Dr. Parikshit Juluri and Ms. Sheyda Kiani Mehr. I am especially grateful for the multiple collaborations in the initial days of my doctoral study, which gave me invaluable insights into manuscript writing

and execute a research project. A big thanks also go to Dr. Mengbai Xiao, who helped me a lot when things seem impossible during my Ph.D. Thank you for taking your valuable time and providing me with all the necessary technical help to deploy the streaming server.

I am thankful to the Computer Science and Engineering department's current and former heads at IIT Guwahati, Prof. Jatindra Kumar Deka, Prof. S. V. Rao, and Prof. Diganta Goswami, for their cooperation and efforts to provide us with departmental resources. I would like to sincerely acknowledge all the department faculty members for their words of wisdom during my conversations with them. I would like to appreciate all the non-teaching staff members of the Department of Computer Science and Engineering. I am especially grateful to Nanu Alan Kachari and Bhriguraj Borah for extending their cooperation in terms of technical support every time I have approached them with a technical problem. I am also thankful to Raktajit Pathak, Pranjit Talukdar, Nava Kumar Boro, Monojit Bhattacharjee, Gauri K. Deori and Prabin Bharali for extending all the official support for the successful completion of my research work.

I would also like to express my heartfelt gratitude to Prof. Nityananda Sarma, who inspired and motivated me to pursue a research career. His invaluable insights into framing a problem statement and executing a research project in the initial days of my research career helped me substantially during my Ph.D. study. I am thankful to Dr. Sanjib Kumar Deka, Dr. Bijoy Chand Chatterjee, Dr. Arindam Karmakar, Dr. Himangshu Sarma and Prakash Chauhan - they spent valuable time sharing their insights on research, life-changing literature they shared with me during tea and helping recommend me for numerous opportunities when my professional life took numerous turns.

A successful Ph.D. would be impossible without great seniors and colleagues. I can boast I had many: Dr. Shirshendu Das, Dr. Shashi Shekhar Jha, Dr. Mayank Agarwal, Dr. Nilkanta Sahu, Dr. Shilpa Budhkar, Dr. Basant Subba, Dr. Lalatendu Behera, Dr. Vishal Deshpande, Dr. Mahesh Patel, Dr. Rakesh Pandey, Dr. Piyoosh P., Dr. Subhrendu Chattopadhyay, Dr. Sukarn Agarwal, Dr. Saptarshi Pyne, Dr. K. Balaprakasa Rao, Dr. Parabattina Bhagath, Manoj Kumar, Basina Deepak Raj, Surajit Das, Sadu Chiranjeevi, Dr. Swarup Ranjan Behera, Dr. Nayantara Kotoky, Dipika Deb, Dr. Abhijit Das, Arunangshu Pal, Dipojjwal Ray, Pawan Kumar Mishra, Arijit Nath, Debabrata Senapati, Swagat Ranjan Sahoo, Alakesh Kalita, Debanjan Roy Choudhury and Suraj Kr. Pandey. Thanks to all of you for your support and made my stay at IIT Guwahati a memorable experience. I especially thank my friends Shilpa, Mayank, Basant, Sukarn, Nayantara, and Dipika for the wonderful times together and for cheering me up after each crucial turning point of my life. You people had changed my perspective to view the world and deeply inspired me to step outside of my comfort zone when negativity invaded my life and took over my thoughts.

During my doctoral study, I also had the opportunity to meet talented and enthusiastic

students: Midhul Verma Vuppalapati, Surabhi Gupta, Ameya Daigavane, Ramkumar Vooda, M. Sai Akhil Teja, Shubham Goel, Shubham Luhadia, Soumak Datta, Anandi R, Jishnu B, Bhagya C, N S Swamys, and Pavan Dasari. It was a great learning experience to mentor these young, motivated minds. Thanks to all of you!

My deepest, sincere gratitude goes to an incredible group of companions and wonderful colleagues, including past and current members of the User-Centric Computing and Networking Lab: Dr. Subrata Tikadar, Dr. Nikumani Choudhury, Sandeep Sarma, Moustafa Najm, Ujjwal Biswas, Papiya Biswas, Md. Shakeel Iqbal Saikia, Nilotpal Biswas, and Dr. Kebede Wakjira Abdalganiy. I wholeheartedly thank them for the timely help they always rendered, for the wonderful times together, for the brainstorming sessions over coffee and dinner parties, and for critically validating (and sometimes invalidating) my ideas when I was uncertain of them. You people have been my family during my stay at IIT Guwahati campus and helped me have a comfortable stay away from my family.

I would also like to thank my wonderful school friends: Sagar Gangula, Basant Sahu, Santosh Kumar Adappa, Surya N Maharana, Rajendra Potnuru and Chaitanya Annabattula for their assistance, encouragement, and inspiration to achieve great heights along the way. They attentively pick up on the slightest signals of my weakness and have in some way, shape, or form pushed me to keep going, to go the extra mile, not to settle. Without their support, I would not have experienced some of the most challenging yet empowering moments of my life.

It goes without saying that this journey would not have been possible without the blessings, unconditional love, persistent support, and profound encouragement of my parents and younger brother. My father did not only raise and nurture me but also taxed himself dearly over the seemingly endless years for my education and intellectual development. My mother has been a source of motivation and strength during moments of despair and discouragement. I thank them for giving me the strength to stand strong by truth, reach for the stars, and chase my dreams. I owe them everything I was able to accomplish in my life. My heartfelt gratitude to my brother for taking care of my parents and dealing with all the hard times with my least assistance. He took the family's responsibility upon himself to ensure that I could concentrate on my Ph.D.

Lastly, I want to offer my regards to all of those who contributed to this dissertation in any respect. I deeply appreciate all the generous support I have received in my life, even down to the smallest contribution.

Guwahati, February 2021 **Hema Kumar Yarnagula**

Abstract

Video streaming applications currently represent a large portion of the Internet traffic, and their relevance is expected to increase in the years to come. This shift is also accompanied by the considerable growth of mobile video traffic. Cisco Visual Networking Index forecasted that the volume of mobile video traffic accounted for 59 percent in 2017 and will account for nearly four-fifth (79 percent) proportion of the total mobile data traffic by the end of 2022 with a 9-fold increase between 2017 and 2022. The rapid growth of video streaming traffic can be rooted back to three different causes, namely widespread multimedia content accessibility, improved user device capabilities, and relevant improvements in video delivery techniques. These relevant improvements have paved the way for HTTP Adaptive Streaming (HAS) that dynamically decides the best quality level for the video segments to download based on the locally perceived network conditions. In light of the wide adoption of the HAS principle, the MPEG consortium has proposed a standard for HAS in 2012, called Dynamic Adaptive Streaming over HTTP (DASH). Of late, DASH has proven to be very effective and becomes the current dominant technology for video streaming over the Internet. Nevertheless, the best-effort Internet was not originally designed to support such bandwidth-intensive video streaming applications. This aspect poses a serious challenge on how to provide a satisfactory viewing experience to the end-users, the so-called Quality of Experience (QoE), under time-varying network conditions, especially in mobile networks. For this reason, the works in this dissertation assess, model and improve the Quality of Experience of Dynamic Adaptive Streaming over HTTP in mobile devices.

First, we investigate how the bitrate adaptation algorithms in DASH influence the end-user QoE, particularly operating under highly fluctuating bandwidth conditions such as mobile networks. To this aim, we perform video QoE assessment (both subjective and objective assessments) of several popular DASH bitrate adaptation algorithms in mobile devices using a trace-driven quality assessment experimental framework. We also present and formally define a set of application-layer objective QoE metrics for the objective quality assessment with an end goal to capture the severity of the metrics on the end-user viewing experience. The objective QoE assessment results demonstrate that bitrate adaptation algorithms that merely utilize either the instantaneous bandwidth estimation or the playback buffer occupancy fares poorly in terms of QoE. Besides, we also observe from the video QoE assessment results that there exists a mutual dependence among the objective QoE metrics. In light of the above, we conclude that due to the trade-offs involved among the influencing QoE metrics, no bitrate adaptation algorithm can lead to better QoE merely by selecting an appropriate bitrate for the next video segments in mobile devices. We also propose a linear parametric QoE model to compute a cumulative QoE score which can be used

for a fair comparison of the bitrate adaptation algorithms. Based on the cumulative QoE score computed, we observe that the bitrate adaptation algorithms following a conservative approach in selecting the video bitrate outperform the others in terms of the cumulative QoE score. We observe that the higher cumulative QoE score for the bitrate adaptation algorithms is mostly attributed to the informed bitrate decision taken with the aim of minimizing the interruptions and bitrate switches while increasing the video bitrate efficiency. We also found that the bitrate adaptation algorithms with an objective to assure a satisfying QoE should try to improve all the QoE metrics rather than focusing on a subset of metrics.

Second, we propose a parametric QoE-estimation model based on multinomial logistic regression (MLR) to predict the overall QoE for a DASH video session. The MLR model depends on a set of objective QoE metrics that can be recorded using lightweight instrumentation at the client and predicts a QoE score that closely resembles the end-user viewing experience for a DASH session. We first compute the dependency using both the quantitative objective assessment results and preferences from the quantitative post-survey stage to identify the influencing objective QoE metrics that affect the QoE as perceived by the end-users. Next, we train the MLR model with the data of session-wise subjective ratings and the corresponding values of objective QoE metrics obtained from subjective QoE assessment to understand the user preferences. The trained MLR model merely takes the measured values of five objective QoE metrics as input and computes the individual probabilities for the response categories (i.e., QoE scores). Subsequently, the response category with the highest probability is predicted to be the QoE score for the given input. We validated the quality-prediction accuracy of the trained MLR model using both hold-out and k-fold cross-validation techniques. We observe a Pearson's linear correlation coefficient of 0.9235 and a root mean square error of 0.416 during the quality-prediction accuracy validation with the hold-out test dataset. We also observe an 83 percent prediction accuracy on an average for the proposed MLR model. The proposed MLR model is compared with three other well-known models from the literature to show that the MLR model gives a good prediction accuracy with reasonably low computational complexity. We also demonstrate the applicability of the model for estimating the QoE for longer video sessions and for any general DASH adaptation algorithm.

Next, we propose an adaptive segment-aware K-push algorithm (termed ASK-Push) to improve the QoE while addressing the request explosion problem leveraging the server-push feature in mobile clients. The proposed adaptive segment-aware K-push algorithm aims to determine an appropriate adaptation push pair (i.e., number of segments and corresponding bitrate level) for the push cycle in DASH/2. The proposed ASK-Push algorithm utilizes variation in the segment sizes to improvise the network throughput estimation and determine a suitable value for the push length in a push cycle. To determine the push length, a buffer model is also considered that ensures not only playback continuity but also addresses the over-push problem. Subsequently, the algorithm takes network throughput estimated using

weighted harmonic mean strategy, segment size information, and current playback buffer occupancy into account to determine an appropriate video bitrate level for the segments in the push cycle. The ASK-Push algorithm uses the stream termination feature of HTTP/2 to cancel the incoming pushed segments in case of a mismatch in the estimated bandwidth. We provide a practical implementation of the DASH/2 streaming prototype with support for the adaptive-push mechanism and low-overhead client implementation for the ASK-Push algorithm. We evaluate the efficiency of our proposed ASK-Push algorithm using an emulated network testbed. The findings from our extensive evaluation demonstrate that the ASK-Push algorithm significantly outperforms existing schemes in terms of a set of objective QoE metrics. The experimental results demonstrate that the ASK-Push algorithm achieved an average of 6%, 67% and 19% improvement for bitrate adaptation efficiency, interruption ratio, and bitrate switching amplitude reward, respectively, compared to other approaches across all the considered experimental scenarios.

Finally, we propose an adaptive segment prefetching (termed ASPF) strategy with an aim to improve the end-user QoE for DASH in Multi-access Edge Computing (MEC) networks. The ASPF strategy dynamically determines the number of segments and their corresponding video bitrate level, leveraging the last-mile radio link throughput prediction and virtual client buffer at the MEC server. Subsequently, it prefetches the video segments while maintaining a sufficient gap ahead of the client's actual segment request progress and store them at the MEC server. Once the client requests are received, they get served directly from the MEC server to reduce the segment access latency and improve the overall QoE for the DASH video session. We provide a practical implementation of the MEC-based streaming prototype with support for the adaptive segment prefetching mechanism. The findings from our extensive evaluation demonstrate that the ASPF strategy significantly outperforms the baseline schemes in terms of overall QoE improvement when compared in the presence of realistic network cross-traffic. The experimental results also demonstrate that the proposed ASPF strategy is able to achieve a significantly higher segment hit ratio at the MEC server for all the bitrate adaptation algorithms considered. The higher prefetched segment hit ratio by the ASPF strategy also demonstrates a better utilization of the MEC-video server backhaul link capacity as compared to the existing baseline schemes. We also observed that all the bitrate adaptation algorithms, when streamed with the proposed ASPF strategy, achieved a significantly higher QoE score compared to the baseline strategies across all the experimental scenarios.

Table of Contents

List of Figures

Page

List of Tables

Page

List of Algorithms

List of Acronyms

Acronym	Expansion
3G	3rd Generation
4G	4th Generation
5G	5th Generation
ARA	Advanced Rate Adaptation
ABMA+	Adaptation and Buffer Management Algorithm
ABR	Adaptive Bitrate
ACR	Absolute Category Rating
AID	Average Interruption Duration
API	Application Programming Interface
AP	Access Point
AR	Auto-Regressive Model
ASK-Push	Adaptive Segment-aware K-push Algorithm
ASPF	Adaptive Segment Prefetching
ASWT	Average Segment Wait Time
BAE	Bitrate Adaptation Efficiency
BB	Buffer-based Algorithm
BBB	Big Buck Bunny
BGP	Border Gateway Protocol
BOLA	Buffer Occupancy based Lyapunov Algorithm
BSAR	Bitrate Switching Amplitude Reward
BSE	Bitrate Switching Events
C2S	Client-to-Server
CAND	Client-and-Network-Assisted DASH
CBM	Cost-aware Buffer Management

CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CI	Confidence Interval
CMCD	Common Media Client Data
CMSD	Common Media Server Data
CS2P	Cross-Session Stateful Predictor
CTA	Consumer Technology Association
D2D	Device-to-Device Communications
DASH	Dynamic Adaptive Streaming over HTTP
DASH/2	Dynamic Adaptive Streaming over HTTP/2
DQ-DASH	Distributed Queuing Theory Bitrate Adaptation Algorithm DASH
EM	Expectation-Maximization
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FR	Full Reference
FSA	Four Segments Ahead
FSMC	Finite-State Markov Channel
FSRA	Fairness-aware Smooth Rate Adaptation
HAS	HTTP Adaptive Streaming
HM	Harmonic-Mean
HMD	Head-Mounted Display
HMM	Hidden Markov Model
HSDPA	High-Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
HTTP/2	Hypertext Transfer Protocol Version 2.0
IF	Influencing Factor
IR	Interruption Ratio
ISD	Initial Startup Delay
ISDR	Initial Startup Delay Reward
ISG	Industry Specification Group
ITU	International Telecommunication Union

LS	Last-Sample
LTE	Long Term Evolution
MDP	Markov Decision Process
MEC	Multi-access Edge Computing
MLE	Maximum Likelihood Estimate
MLR	Multinomial Logistic Regression
MMDBMS	Multimedia Database Management System
MNO	Mobile Network Operator
MOS	Mean Opinion Score
MPC	Model Predictive Control
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MS-Stream	Multiple-Source Streaming over HTTP
NAT	Network Address Translation
NFV	Network Function Virtualization
NQQ	Normalized Quantization Stepsize
NR	No Reference
OSMF	Open Source Media Framework
P-GW	Public Gateway
PBHR	Prefetched Byte Hit Ratio
PDF	Probability Distribution Function
PID	Proportional-Integral-Derivative
PLCC	Pearson's Linear Correlation Coefficient
PLR	Packet Loss Ratio
PoP	Points of Presence
PSHR	Prefetched Segment Hit Ratio
PSNR	Peak Signal-to-Noise Ratio
QAAD	QoE-enhanced Adaptation Algorithm over DASH
QK-Push	QoE-driven Adaptive K-Push Mechanism
QoE	Quality of Experience
QoS	Quality of Service

QP	Quantization Parameter
QUETRA	QUEuing Theory-based Rate Adaptation
QUIC	Quick UDP Internet Connection Protocol
RAN	Radio Access Network
RB	Rate-based Algorithm
RBPS	Red Bull Playstreets
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNIS	Radio Network Information Service
RR	Reduced Reference
RTT	Round-Trip Time
SAND	Server-and-Network-Assisted DASH
SARA	Segment-Aware Rate Adaptation
SDN	Software-Defined Networking
SSIM	Structural Similarity Index Measure
SSL	Secure Socket Layer
SVC	Scalable Video Coded
TCP	Transmission Control Protocol
TFDASH	Throughput-Friendly DASH
ToS	Tears of Steel
UPD	User Datagram Protocol
URL	Uniform Resource Locator
VCI	Video Continuity Index
VM	Virtual Machine
VMAF	Video Multimethod Assessment Fusion
VoD	Video-on-Demand
VQM	Video Quality Metric
VR	Virtual Reality
VRT	Vlaamse Radio- en Televisieomroeporganisatie

List of Symbols

Symbol	Description
K	Total number of video segments downloaded by the client.
$\mathbb{R}$	Available video adaptation set that includes all the video bitrate levels.
R_k	Video bitrate of the k^{th} ($k = \{1, ..., K\}$) video segment downloaded.
L_k	Quality level of the k^{th} video segment downloaded.
$max_j R_j$	Video bitrate of the highest available quality level from $\mathbb{R}$.
$B(t)$	Playback buffer occupancy at time t (represented as discrete instants of observation in seconds).
BW_T	Bottleneck bandwidth computed by the client across the K segments.
t_s	Start time of an interruption.
t_e	End time of an interruption.
ΔT	Total duration of the video session (including both play and pause time).
ΔT_{Int}	Total time the video playback is paused due to interruptions.
ISD^{MAX}	The maximum startup delay observed across all the experiments for a given scenario.
β_j	The intercept of the multinomial log-odds for the response categories (i.e., $j = 1, 2, ..., 4$) .
x_1	The explanatory independent variables associated with BAE.
x_2	The explanatory independent variables associated with IR.
x_3	The explanatory independent variables associated with BSAR.
x_4	The explanatory independent variables associated with VCI.

x_5	The explanatory independent variables associated with ISDR.
$\beta_{j,k}$	The regression coefficient associated with the jth probability outcome and kth independent variable.
$\beta_{1,1}$ to $\beta_{4,1}$	The regression coefficients associated with BAE.
$\beta_{1,2}$ to $\beta_{4,2}$	The regression coefficients associated with IR.
$\beta_{1,3}$ to $\beta_{4,3}$	The regression coefficients associated with BSAR
$\beta_{1,4}$ to $\beta_{4,4}$	The regression coefficients associated with VCI.
$\beta_{1,5}$ to $\beta_{4,5}$	The regression coefficients associated with ISDR.
κ	Actual number of segments selected for the push cycle.
$\hat{b}_\kappa$	Video bitrate selected for a push cycle; same bitrate for all the κ segments.
n	Number of segments downloaded so far in a push cycle.
W_{n+i}	The segment sizes for κ segments i.e., $\{w_{n+1}, w_{n+2}, \ldots, w_{n+\kappa}\}$ with video bitrate $\hat{b}_\kappa$, respectively.
r^{min}	Lowest available video bitrate in $\mathbb{R}$.
r^{max}	Highest available video bitrate in $\mathbb{R}$.
φ	Segment play duration.
B_{curr}	Current playback buffer occupancy at time t.
B_{res}	Buffer reservoir for the fast start phase.
B_{crit}	Critical buffer threshold.
B_{max}	Maximum buffer threshold.
K_{max}	Maximum number of video segments allowed to push during a push cycle; to avoid the over-push problem.
K_{min}	Minimum number of segments for the current push cycle required to to fill the buffer beyond the B_{crit} threshold.
HM_n	Weighted harmonic mean throughput estimated on last n segment downloads.
V_l	Total video session length.
V_d	Video content downloaded so far (i.e., $n \times \varphi$).
g^*	Number of segments pushed (i.e., group size) during the previous push cycle.
b^*	Segment bitrate chosen in the previous push cycle.

B_{esti}	Anticipated playback buffer occupancy estimated using κ and $\hat{b}_\kappa$ values as input.
δ	The wait time before deciding appropriate κ and $\hat{b}_\kappa$ for the next push cycle.
$BW_r(t)$	Actual measured bandwidth value from the beginning of the push cycle (i.e., t_0).
$BW_p(t)$	Predicted bandwidth value for the push cycle.
$BW_{Diff}(t)$	Bandwidth mismatch between $BW_r(t)$ and $BW_p(t)$ discovered by the ASK-Push approach.
M_s	Denotes a given set of features from all possible feature combinations for any session s.
M_s^*	Denotes the set of features that are mapped to each session s and yields the lowest absolute normalized prediction error.
$Agg(M_s^*, s)$	State-transition behaviour in each cluster.
W_t	The random variable representing the network throughput at time period t.
w_t	The actual downlink throughput measured from the RNIS module.
$\hat{W}_t$	The predicted value of W_t.
$\mathcal{X}$	The set of all possible discrete states.
N	The number of states and is given by $\lvert\mathcal{X}\rvert$.
X_t	Random hidden state variables and the network throughput W_t evolves according $X_t \in \mathcal{X}$.
Π	Denotes the probability distribution vector for X_t.
$\mathbf{P}$	The transition probability matrix.
$N(\mu_x, \sigma_x^2)$	The mean and its standard deviation of the Gaussian distribution of network throughput for a state in the HMM.
$e(w_t)$	The emission probability vector.
$f(\cdot)$	The Gaussian probability distribution function.
$\circ$	Denotes entry-wise multiplication of the two vectors (also called as Hadamard product).

CHAPTER

1

Introduction

In recent years, we have witnessed the emergence of high-speed mobile data access networks, such as UMTS, LTE, WiMAX, and other similar standards, increasing the data rates exponentially year after year. Nevertheless, the end-users have had no problem finding ways to utilize the available Internet access bandwidth and go beyond just web-browsing on mobile devices. Of late, mobile video streaming is one of the primary applications making use of the improved data rates to deliver high-quality video(s). Besides, with the advancements of device technology, there has been a significant change in the way people consume multimedia content using mobile devices such as smartphones and tablets. This shift is also accompanied by the considerable growth of mobile video traffic and mobile video streaming is already the most significant IP traffic generator throughout the Internet. Cisco Visual Networking Index forecasted that the volume of mobile video traffic accounted for 59 percent in 2017 and will account for nearly four-fifth (79 percent) proportion of the total mobile data traffic by the end of 2022 with a 9-fold increase between 2017 and 2022 [5]. This is mostly due to an increase in the popularity of HTTP-based video streaming services (such as YouTube, Netflix, and Hotstar) and the rapid adoption of mobile devices.

The rise in HTTP-based video streaming services popularity is mainly due to the following reasons: (i) existing Internet infrastructure supports HTTP and traditional stateless web servers, localized edge caches provided by the content distribution networks (CDNs) can be used by the service providers to stream the videos, (ii) HTTP is firewall-friendly and can effortlessly traverse the network middleboxes that include network address translators,

traffic shapers, application accelerators, and transparent Web proxy caches, etc., and (iii) unlike Real-Time streaming Protocol (RTSP) and Real-Time Messaging Protocol (RTMP), Hypertext Transfer Protocol (HTTP) is a stateless protocol. Due to the stateless nature of HTTP servers, the functionalities required for smooth streaming of video content can be implemented at the streaming client.

Dynamic Adaptive Streaming over HTTP (DASH) [6, 7] is an adaptive bitrate streaming solution and retains all the advantages of traditional progressive streaming. Besides, DASH overcomes the several limitations of the traditional progressive streaming, such as wastage of bandwidth in aggressive buffering and lack of adaptation to bandwidth variations [8]. DASH enables the streaming of video content from conventional HTTP web servers. It also allows the streaming client to adapt the video segment bitrate according to the current network conditions and user preferences. Currently, DASH is widely used on the Internet by several leading video content providers such as YouTube, Netflix, and Vimeo for delivering video(s) to both wired desktop and mobile users.

In DASH, the raw video file is encoded into several representations/bitrate levels. Each video representation fundamentally varies in encoding rate, display resolution, format, etc. Furthermore, each of the newly encoded video files is split into multiple smaller chunks called segments/fragments. Each video segment has a fixed playback duration (typically 1-15 seconds). A Media Presentation Description (MPD) file to store the video metadata (such as video playback duration, set of available representations, codec used, segment duration, and unique base URL for each segment) is prepared for each DASH videos. These individual video segments with a unique URL and the MPD file are hosted on conventional HTTP web servers. DASH clients follow a client-pull paradigm that requests the video segments which might belong to different representations from the streaming server by adapting to time-varying network conditions and several other end-to-end system factors. When the client (media player) requests the video, the MPD file is sent to the client to deliver all the information required to fetch the desired segments during the video session. Typically, the DASH client media player has two key modules, namely the media module and the adaptation module. The media module is responsible for rendering the video frames from the playback buffer, while the adaptation module is responsible for selecting the appropriate representation for the next segments. The *bitrate adaptation algorithm*

used by the adaptation module could either be conservative or aggressive in selecting the video bitrate levels for the future segments. Hence, with highly dynamic network conditions, it would influence the end-user viewing experience and can also have degradation in the quality of experience [9].

Quality of experience (QoE) is defined as the overall acceptability of an application or service, as perceived by the end-user [10]. Overall acceptability of the video streaming service is typically influenced by several factors such as available network bandwidth, client device characteristics, and server load. Of late, the market for both mobile network service and video service providers is highly competitive and fast-growing. The evolving world of video streaming not only offers opportunities but also poses several challenges for both the mobile network service and video service providers. In particular, adaptive video streaming in resource-constrained mobile devices that operate on dynamically changing and especially low-bandwidth mobile networks (particularly true for many developing countries) poses quite a different challenge for content providers [11, 12]. It is evident that these service providers who can continuously improve their competitive advantages over their competitors can only continue in this highly competitive market. To this end, the ultimate goal of every video service provider is to stream the highest possible video quality while making efficient use of the limited network resources. Since the end-user has a wide range of streaming service options to choose from, all the efforts will be ineffectual if the quality of video services fails to ensure a satisfactory viewing experience to the paying subscribers. This could also potentially lead to the decline in the popularity of the video services and revenue.

1.1 Motivation for the Research Work

In recent years, several bitrate adaptation algorithms have been proposed in the research literature [8, 13] to deal with the challenge of assuring an acceptable QoE to the end-users. Although a substantial amount of work on bitrate adaptation algorithms for DASH has been undertaken, to the best of our knowledge, their performance for the delivery of DASH video streams in mobile devices has not been well researched in the existing literature. Therefore, the objective of this thesis is to analyze the problem of QoE-centric DASH video delivery in mobile devices from an end-user point of view, in particular, to design control actions to improve the end-user QoE for DASH streams in mobile devices. In light of the above, we

identify the following challenges for QoE-centric DASH delivery in mobile devices:

Challenge #1: ***Assessing end-users' QoE for popular DASH bitrate adaptation algorithms in mobile devices, particularly operating under fluctuating network conditions.***

In a typical HTTP adaptive streaming environment, multiple clients simultaneously access the video content from a remotely located video streaming server. Often, these multiple clients have to share a single medium (i.e., network link) to download the video segments. In such scenarios, the competition for the single shared medium could result in issues concerning fairness among the streaming clients. This means the presence of an aggressive streaming client has a negative impact on the streaming performance of all other clients sharing the medium. This fairness issue particularly becomes apparent when multiple DASH clients share single or multiple bottleneck network links. Akhshabi et al. in [14] and [15] have experimentally evaluated the bitrate adaptation mechanisms of popular adaptive streaming players to identify the noteworthy inefficiencies and their behavior when they compete for bandwidth on a bottleneck link. Their analysis has uncovered that the issues concerning fairness are not due to the Transmission Control Protocol (TCP) dynamics but mainly arise from the aggressiveness of the bitrate adaptation algorithms employed, as they decide on the actual video bitrate for the segment to download. When multiple streaming clients download video segments on the shared bottleneck link, the temporal overlap of the clients' activity-inactivity periods due to the nature of DASH bitrate adaptation algorithms could result in inaccurate bandwidth estimation.

In particular, when multiple clients compete for a shared bottleneck link, the inaccurate bandwidth estimation by the streaming client subsequently affects the video bitrate selection. This could also lead to either frequent playback interruptions or unnecessary bitrate switching events during the video session. The frequent playback interruptions and bitrate switching events are easily perceived by the viewer and are known to have the worst influence on the end-users viewing experience. Thus, this could significantly degrade the end-user QoE. The problem is further aggravated mainly due to the unmanaged nature of DASH bitrate adaptation algorithms in mobile clients that operate in networks with high bandwidth fluctuations, delay variation and high packet loss rate.

Challenge #2: ***Predicting QoE as perceived by the end-user during the video playback.***

In general, the QoE perceived by the end-user is captured by performing a subjective QoE assessment. In subjective assessment, a participant is asked to watch the test video clip(s) in a controlled laboratory environment. Then the participant is asked to grade the perception based on their viewing experience. The subjective QoE assessment is an important and very active research field in the broader domain of QoE optimization [16]. Despite the importance and ability to provide a better understanding of video quality experienced by the end-users, the subjective QoE assessment has inherent drawbacks. Unfortunately, subjective QoE assessment is both time-consuming and resource-consuming. This makes the subjective QoE assessment impractical for real-world applications as obtaining the feedback directly from the viewer is not easy. Furthermore, conducting large-scale subjective measurements to obtain the statistically meaningful subjective perception is an onerous job.

In recent years, as an alternative solution to the subjective QoE assessment approach, researchers have envisaged QoE models to predict/estimate the overall viewing experience. In practice, the QoE prediction models, especially for mobile video streaming, can be used by a variety of stakeholders like mobile network operators, video streaming service/platform providers, and regulators. The QoE prediction models in the literature, based on the type of input and the location where the input information is extracted [4], can be broadly classified into four categories: (i) media-layer models, (ii) bitstream models, (iii) parametric models, and (iv) hybrid models. In recent times, objective metric-based parametric QoE models have been widely used in the literature [17, 18, 19, 20]. The objective metric-based parametric QoE models aim at correctly determining a quality score making use of the set of industry-standard objective QoE metrics [21] such as playback interruptions, bitrate switches, average video bitrate, etc. Nevertheless, the parametric QoE models in the research literature have the following limitations [4, 22].

- First, for end-point quality monitoring purposes, a parametric quality-estimation model should consider the clear insight of the user-related factors and individual preferences.
- Second, the QoE model must also consider the previous experience, which may be in the form of trained models, together with real-time application-level data collected

from the client during service usage to improve the accuracy of the quality score prediction.

- Third, the QoE model should have lower computational complexity to make it suitable for devices with constrained power and processing capabilities such as mobile phones.

- Finally, the parametric QoE model that takes into account only application-level parameters (i.e., objective QoE metrics [21]) will be reasonably simple to implement in mobile devices.

In the wake of the limitations mentioned above, there exists a serious need for the development of an accurate QoE estimation model that tries to take into account the objective QoE metrics, a clear insight of the user-related factors and individual preferences to accurately predict the overall perceived QoE, which closely resembles the actual user viewing experience. This would also be helpful for the service providers to optimize their adaptation algorithm with an aim to improve the overall QoE during a video session.

Challenge #3: ***Improving video QoE of DASH while mitigating the request explosion problem and addressing the over-push problem in mobile devices.***

Dynamic Adaptive Streaming over HTTP is widely used by several leading video content providers, such as YouTube, Netflix, and Vimeo, for delivering video(s) to both desktop and mobile users. However, HTTP/1.1 imposes additional overhead in terms of the request messages used to fetch the segments (termed *request explosion*), apart from other challenges (due to protocol design) such as head-of-line blocking and fat request/response headers [23]. The request message overhead mostly increases the end-to-end latency as it takes more than one RTT before the next segment is downloaded. The overall latency could be more prominent for mobile networks due to constrained uplink bandwidth, leading to degraded end-user QoE [9].

HTTP/2, standardized in 2015, promises significant performance improvements over HTTP/1.1, especially with the server-initiated push mechanism feature called *Server Push*. The *Server Push* feature allows the streaming server to push multiple additional segments in response to the client's primary segment request. It mitigates the *request explosion* problem and reduces the latency incurred in downloading segments, especially for mobile clients.

Leveraging the *Server Push* feature of HTTP/2, a few researchers designed K-push schemes for both low latency live streaming [23, 24, 25] and stored video streaming [26, 27]. In [26], the authors investigated the benefits of *fixed K-push scheme* for Dynamic Adaptive Streaming over HTTP/2 (DASH/2). The client leveraging the fixed K-push scheme determines the appropriate bitrate level ($\hat{b}_\kappa$) for the fixed number of segments (κ) to be pushed during every push cycle. Subsequently, the client sends the request with the *Push Directive* (consists of both κ and $\hat{b}_\kappa$ values) to the video server. The server initiates a new push cycle, where κ segments of bitrate level $\hat{b}_\kappa$ are pushed without a separate request for each segment. Their experimental results have uncovered that a fixed K-push scheme is able to improve both network utilization and adaptability significantly. However, fixating the push cycle's group size (i.e., κ) could superfluously fill the playback buffer and result in an over-push problem. To overcome the over-push problem, the works in [26, 27] propose to determine a suitable adaptation push pair, i.e., both bitrate level and the number of segments, at runtime. Nevertheless, It is evident that the number of segments and the choice of corresponding bitrate level in a push cycle are not independent. Selecting a fewer number of segments in the push cycle could lead to increased request explosion. On the contrary, a lower bitrate level for the video segments in a push cycle could result in notable degradation in end-users QoE. Hence, the delivery of video segments using DASH/2 is not straightforward. In addition, it is vital to appropriately determine both the group size and respective segment bitrate level of a push cycle to ensure acceptable QoE to the end-users.

Although DASH video segments typically have the same playback duration (ranging from 1-15 seconds), the segment sizes even across the same bitrate level could vary significantly. The DASH/2 K-push adaptation algorithms in the literature consider either the estimated network throughput, playback buffer occupancy, or a combination thereof to decide the suitable adaptation push pair for the next push cycle. However, they do not take into account the variation in the segment size and its impact on the estimated throughput and/or buffer filling rate. Hence, they could fail to accurately forecast the time required to download the κ segments of a specific bitrate level for a push cycle, particularly when operating under dynamic and low bandwidth conditions such as in mobile networks. This could even result in frequent playback interruptions (due to playback buffer depletion) that would subsequently lead to the selection of lower bitrate levels for the push cycles

in the near future. Furthermore, the K-push schemes in the literature do not take into consideration the impact on metrics influencing end-users QoE, such as video bitrate quality, playback continuity, bitrate switching, and initial startup delay. The work in [28] studied the QoE with different bitrate adaptation algorithms to demonstrate the mutual dependence among the major influencing QoE metrics. Thus, a K-push scheme with the goal to merely maximize a subset of the influencing QoE metrics might be unable to adapt and provide a satisfactory viewing experience to the end-user under highly dynamic scenarios. Therefore, it is required to design an adaptive K-push algorithm that considers segment size information to estimate the time required to download the next κ segments in a push cycle. In addition, the K-push algorithm is required to avoid playback buffer depletion while ensuring the best possible video bitrate is selected for the segments based on the network and streaming conditions so that they can improve the overall end-user QoE in mobile clients.

Challenge #4: ***Improving end-users QoE leveraging segment prefetching strategy in Multi-access Edge Computing networks.***

In recent times, the rapid growth rate in mobile video streaming traffic [5] and bandwidth demand create immense pressure on mobile network operators. Besides, with the surge in the number of mobile users requesting video content over the Internet, the video service providers also started observing complications related to their services' performance and scalability. Hence, DASH video playback sessions with frequent video bitrate oscillation and interruptions could severely degrade the end-user QoE and translate into a decline in popularity for the video service providers. In the light of this, the video service providers with an ultimate goal to provide the expected level of QoE to end-users have moved beyond traditional client-server streaming and embraced content delivery networks (CDN) for improving their video content distribution. The video content pre-cached at the CDN servers (usually located near the network edge) significantly reduces the video segment access latency. Thus, the end-users are likely to get improved video QoE. Nevertheless, there exist several limitations that inhibit the video service provider from using the CDN approach for the delivery of multimedia content [29, 30]. Some of these limitations mainly include lack of flexibility, lack of control and limited access, and limited points of presence (PoP).

In the wake of these problems, *Multi-access Edge Computing* (MEC) was envisaged

as an alternative solution to the CDN approach by the standards organization European Telecommunications Standards Institute (ETSI) in 2014 [31]. The MEC paradigm enables mobile network operators (MNO) and multimedia content providers to collaborate directly at the mobile network edge. The MNO provides the virtualized hardware resources that include computing power, storage, and communication resources to the video service provider on a rented basis. The video service provider can employ their own content intelligence (e,g., prefetching or caching) as a virtual network function at the mobile network edge. The video service provider can also manage the content manipulation policies based on the real-time context of networks and/or end-users while placing the video segments at the source streaming server on a per-URL basis.

Of late, several research works have proposed DASH video segment prefetching at the network edge for improving end-user video QoE [32, 33, 34]. Nevertheless, these existing prefetching schemes in the literature follow a rigid policy, i.e., only prefetch either one video segment at a time or a fixed number of segments ahead of time. On the contrary, the DASH client typically decides the video segment bitrate based on the dynamic network conditions and current playback buffer occupancy prior to sending the segment request(s). Thus, prefetching a fixed number of video segments does not perform well, especially in the radio access network (RAN) environment with highly fluctuating network conditions. This could also result in frequent segment misses at the network edge. The segment miss at the network edge entails a higher risk of falling back to the traditional client-server approach for downloading the client-requested segment with significantly higher access latency. Thus, this could negatively impact the end-user viewing experience and degrade the video QoE significantly. In addition, prefetching too many video segments could also result in an injudicious utilization of the limited storage resources of the MEC network edge, especially if the segments are stored for a longer duration before getting removed due to a miss. It is therefore required to develop a segment prefetching scheme to dynamically determine the number of segments and their corresponding video bitrate for prefetching with an aim to address the issues as mentioned above and improve the end-user QoE in mobile clients.

1.2 Contributions of the Thesis

The research challenges for the QoE-centric DASH video delivery in mobile devices, as presented in Section 1.1, are tackled in the remainder of the thesis. We provide a synopsis of the research contributions addressing the research challenges within this thesis are as follows.

1.2.1 Video Quality of Experience Assessment of DASH Bitrate Adaptation Algorithms in Mobile Devices

We perform video QoE assessment including both subjective and objective assessment of several popular DASH bitrate adaptation algorithms in mobile clients (presented in Chapter 3, addressing Challenge #1). We implement a video quality assessment experimental framework with a trace-driven network testbed to emulate realistic DASH streaming in mobile devices. We implement nine popular bitrate adaptation algorithms[1] and the low-overhead client implementation is based on an open-source `dash.js` framework [35]. We perform controlled subjective quality experimentation to capture the end-users' viewing experience for DASH video streams in mobile devices. In each experiment, the test participant has participated in several DASH video sessions with various network trace profiles and adaptation algorithm combinations. We present a detailed analysis of the obtained qualitative subjective assessment results (i.e., subjective rating) from the subjective experiments.

We present and formally define a set of novel objective QoE metrics for the objective quality assessment, which were adapted from the widely accepted objective video quality metrics reported in the literature [8, 21]. The objective QoE metrics sufficiently capture the severity of the objective video quality metrics instead of merely measuring them as a numerical value. We then perform extensive experiments to collect the quantitative objective assessment results for the bitrate adaptation algorithms and network trace profile combinations, using lightweight client-side instrumentation in the background. Apart from this, we also record the quantitative objective assessment results during the subjective assessment. We perform a detailed analysis of the results obtained from the objective QoE assessment to show how the popular bitrate adaptation algorithms fare in the presence of realistic mobile network scenarios.

[1] `https://github.com/hemakyarnagula/DASHAlgorithmCode/`

At the end of each subjective experiment, we collect the quantitative post-survey measures using an exit survey. The aim of the exit survey is to capture the test participants' preference/expectation for the objective QoE metrics in a quantitative manner. We then compute the non-negative weighing parameter values that correspond to the objective QoE metrics to determine the ranking. The larger the value of the non-negative weighing parameter, the higher is the influence of the corresponding metric on the user's viewing experience. In a bid to design a quantitative measure for video QoE, we propose a linear parametric model to compute a cumulative QoE score. The cumulative QoE score is used for a fair comparison of the bitrate adaptation algorithms. The proposed linear parametric QoE model considers the observed objective QoE metrics values and their priorities (i.e., corresponding non-negative weighing parameter) to compute the QoE score. For the video QoE assessment, we implement nine popular bitrate adaptation algorithms from three different categories. We perform the QoE assessment with three video sequences and three bandwidth trace profiles. The key observations from the results are as follows.

- From subjective QoE assessment, we observe that the bitrate adaptation algorithm that considers both throughput and buffer occupancy to make an informed bitrate decision outperforms others in terms of the mean opinion score. The bitrate adaptation algorithm that considers both estimated throughput and buffer occupancy improves the mean opinion score by an average of 12% compared to others across all trace-video scenarios.

- We compute a cumulative QoE score to assess the bitrate adaptation algorithms' overall performance using the proposed linear parametric QoE scoring model. The segment-aware rate adaptation algorithm outperforms all the other bitrate adaptations algorithms in terms of QoE score under most of the trace-video scenarios considered.

1.2.2 Predicting Quality of Experience in DASH Using Parametric Multinomial Logistic Regression Model

We propose a parametric multinomial logistic regression (MLR) model for video QoE estimation (presented in Chapter 4, addressing Challenge #2). The proposed MLR model predicts both short-term and long-term QoE scores as perceived by the end-user for a DASH

video session. We first compute the dependency using the quantitative objective results captured during subjective experiments and preferences from the quantitative post-survey stage in order to understand the objective QoE metrics influence on the user's viewing experience. With this step, we only determine the set of objective QoE metrics that satisfy the global null hypothesis to train the proposed MLR model. We then train the proposed MLR model using both the subjective assessment results and the corresponding objective assessment results (obtained in Chapter 3). We compute the intercepts and regression coefficients for the independent variables (i.e., the five objective QoE metrics) using an iterative procedure for the training module. Subsequently, the training module models the individual probabilities (for the response variables) using the multinomial log-odds. The trained model predicts an overall QoE score for a video stream by merely taking a set of five objective QoE metrics values as input. We validate the model's prediction accuracy using both the hold-out and k-fold cross-validation techniques that also demonstrate the applicability of the model across different scenarios. We use the proposed model to demonstrate the performances of a few popular DASH adaptation algorithms in terms of the QoE score predicted from the objective QoE metrics. We compare the proposed model with three other well-known models in the literature, which shows a better prediction of QoE with a reasonably low computational complexity, suggesting its suitability for online use. Finally, we demonstrate the proposed model's applicability for videos of longer duration (around 60 minutes) and for both short-term and long-term QoE score predictions. The key observations from the results are as follows.

- We observe a Pearson's linear correlation coefficient (PLCC) of 0.9235 and a root mean square error of 0.416 for the proposed QoE-estimation model during the accuracy validation phase. The proposed QoE-estimation model achieves an 83 percent prediction accuracy with the validation data.

- The proposed QoE-estimation model achieves a mean prediction accuracy of 81 percent when validated with the K-fold cross-validation technique.

- The proposed QoE-estimation model achieves a high PLCC value of 0.83056 as compared to the minimum performance benchmark (i.e., "PLCC $\geq$ 0.70") even when trained with a training set consisting of 90 samples.

- The proposed QoE-estimation model improves the Pearson's linear correlation coefficient by 3% and reduces the root mean square error by 15% when compared to the existing QoE log metric [36], [37] in the literature.

- The proposed QoE-estimation model exhibits a reasonably low computational complexity, i.e., less than 0.1 second, to predict the quality score, suggesting its suitability for online use in resource-constrained mobile devices.

1.2.3 Towards Improved QoE with Adaptive Segment-aware K-Push Algorithm for Dynamic Adaptive Streaming over HTTP/2

We propose an adaptive segment-aware K-push algorithm (termed ASK-Push) to determine the appropriate adaptation push pair (κ, and $\hat{b}_\kappa$) for DASH/2 with an objective of improving QoE in mobile clients (presented in Chapter 5, addressing Challenge #3). The proposed ASK-Push algorithm utilizes segment size information to estimate the time required to download the next κ segments in a push cycle. Subsequently, it performs either uphill adaptation or downhill adaptation to decide the appropriate video bitrate for the segments in the next push cycle. The proposed algorithm leverages the buffer model with three buffer thresholds to alleviate the segment over-push problem. We propose push cycle termination using the HTTP/2 stream termination feature if a bandwidth mismatch is discovered beyond a threshold while downloading the video segments. The push cycle termination will subsequently cancel out all the upcoming pushed segments by leveraging the stream termination feature. We provide a practical implementation of the DASH/2 streaming prototype in `Jetty` [38] with support for the adaptive-push mechanism. We provide a low-overhead client implementation for the ASK-Push algorithm[2] and is based on an open-source `dash.js` framework [35]. We demonstrate the ASK-Push algorithm efficiency by performing systematic objective QoE assessment in an HSDPA/3G trace-driven network testbed. The key observations from the experimental results are as follows.

- The bitrate adaptation efficiency[3] for the ASK-Push algorithm improves by an average of 7% and 4% as compared to the DASH2M [27] scheme when streamed with the Big

[2] `https://github.com/hemakyarnagula/ASK-PushAlgorithmCode/`

[3] `Bitrate adaptation efficiency metric is defined as the average video bitrate played with respect to the bottleneck bandwidth and the highest available video representation bitrate. Bitrate adaptation efficiency is formally defined in Section 3.5.1: Eq. (3.2).`

Buck Bunny and Red Bull PlayStreets video sequence, respectively.

- The interruption ratio[4] is also improved by an average of 78% and 64% when streamed with the Big Buck Bunny and Red Bull PlayStreets video sequence, respectively. This is mostly because the ASK-Push algorithm follows a conservative approach in selecting the video bitrate to avoid frequent playback buffer depositions.

- The average interruption duration reduces by an average of 84% and 65% using the proposed ASK-Push approach when streamed with the Big Buck Bunny and Red Bull PlayStreets video sequence, respectively. The average interruption duration reduction is because the ASK-Push approach strategically manages the playback buffer occupancy in the desired range leveraging three pre-configured buffer thresholds. This also improves the video continuity index metric (i.e., probability of avoiding the interruptions) by an average of 2% across all the experimental scenarios considered.

- We compute the QoE score using the QoE-estimation model (presented in Chapter 4) to predict the overall performance of the algorithms, including the ASK-push algorithm. The proposed ASK-Push strategy exhibits an 8% higher QoE score on an average compared to the DASH2M [27] scheme in the literature.

1.2.4 An Adaptive Segment Prefetching Strategy for QoE-Improved DASH in Multi-Access Edge Computing Networks

We propose an adaptive segment prefetching (termed ASPF) strategy to dynamically determine the number of segments and their corresponding video bitrate for prefetching with an aim to improve the end-user QoE in mobile clients (presented in Chapter 6, addressing Challenge #4). We consider a Hidden Markov Model-based throughput predictor, based on first-order Markovian assumption, for last-mile radio link throughput prediction at the MEC server. We propose a virtual client buffer inference mechanism that simulates a virtual client playback buffer (referred to as a mirrored buffer) at the MEC server leveraging the explicit feedback signaling from the client. The adaptive segment prefetching decision engine

[4] Interruption ratio metric estimates the severity of interruptions for the entire streaming session and is defined as the ratio of the number of playback interruptions recorded and the number of segments downloaded during a streaming session by the streaming client. Interruption ratio is formally defined in Section 3.5.1: Eq. (3.3).

maintains a per-user per-session thread and starts with zero assumptions about the client status. The prefetching decision engine at the MEC server runs ahead of the client request progress. The prefetching decision engine employs a bitrate adaptation algorithm similar to that of the client to decide the number of segments and their corresponding video bitrate to prefetch and store the segments at the MEC server. The prefetching decision engine also carefully addresses the trade-off between the video QoE degradation and injudicious utilization of the MEC server resources by dynamically deciding the number of segments to prefetch. We provide a practical implementation of the MEC-based streaming prototype[5] with support for the adaptive prefetching mechanism. We implement the proposed MEC-based prefetching scheme in `Jetty` [38], an open-source server implementation. We evaluate and analyze our proposed ASPF strategy's efficiency using a set of LTE throughput variability traces, various backhaul link scenarios and low-overhead client implementation[6] of a few popular DASH bitrate adaptation algorithms. The detailed findings from our extensive evaluation characterize the gains of the proposed ASPF strategy compared to state-of-the-art baseline solutions in the presence of realistic network cross-traffic. The key observations from the experimental results are as follows.

- The proposed ASPF strategy improves bitrate adaptation efficiency for the segment-aware rate adaptation algorithm by an average of 23% under the 500ms backhaul scenarios.

- The proposed ASPF strategy with the segment-aware rate adaptation algorithm able to achieve an average of 83% segment hit ratio at the MEC server under the 500ms backhaul scenarios.

- We also observe an improvement of 25%, 31% and 34% for bitrate adaptation efficiency with proposed ASPF strategy and segment-aware rate adaptation algorithm under 300ms, 200ms and 100ms backhaul scenarios, respectively.

- The proposed ASPF strategy exhibits 59%, 44% and 35% increase for segment hit ratio with segment-aware rate adaptation algorithm under 300ms, 200ms and 100ms backhaul scenarios, respectively. Similarly, it also exhibits 72%, 55% and 48% increase

[5] `https://github.com/hemakyarnagula/MECPrototypeCode/`
[6] `https://github.com/hemakyarnagula/DASHAlgorithmCode/`

for segment hit ratio with buffer occupancy-based Lyapunov algorithm under 300ms, 200ms and 100ms backhaul scenarios, respectively.

- We compute the QoE score using the QoE-estimation model (presented in Chapter 4) to predict the overall performance for different bitrate adaptation algorithms. The ASPF strategy with the segment-aware rate adaptation algorithm [39] exhibits 6%, 12%, and 11% higher QoE score as compared to buffer occupancy based lyapunov algorithm [37] under 300ms, 200ms, and 100ms backhaul scenarios, respectively. Similarly, with buffer occupancy-based Lyapunov algorithm, our ASPF strategy exhibits 24%, 18%, and 12% higher QoE scores as compared to the adaptation and buffer management algorithm [40] under 300ms, 200ms, and 100ms backhaul scenarios, respectively.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 begins by presenting background on the architecture of DASH streaming systems. Then, the concept of quality of experience is introduced, and common methods for subjective and objective QoE assessment are described. This is followed by a brief discussion on DASH/2 and Multi-access Edge Computing. Finally, the chapter provides a state-of-the-art review on bitrate adaptation algorithms for DASH, QoE estimation models for HTTP adaptive streaming, and prefetching and caching strategies for DASH.

Chapter 3 presents video QoE assessment of popular DASH bitrate adaptation algorithms in mobile devices. The chapter presents a quality assessment experimental framework that was developed to perform the video QoE assessment in mobile devices. The chapter also provides an overview and challenges in the ope-source `dash.js` framework and presents the modifications incorporated into the `dash.js` code for integrating the bitrate adaptation algorithms. Then, the chapter presents a detailed analysis of the bitrate adaptation algorithms performance in mobile devices operating under highly variable bandwidth conditions.

Chapter 4 presents a parametric multinomial logistic regression model to predict both short-term and long-term QoE scores as perceived by the end-user for a DASH video session. The chapter provides a detailed description of the model training procedure. Then, the

chapter presents a discussion on the proposed model accuracy validation results and the model's applicability for longer duration videos.

Chapter 5 discusses the work done to address the request explosion problem and improve QoE with an adaptive segment-aware K-push algorithm for Dynamic Adaptive Streaming over HTTP/2. The chapter presents the full description of the adaptive segment-aware K-push algorithm to make an informed adaptation push pair decision for DASH/2. The chapter also provides a detailed discussion of the objective QoE assessment results to demonstrate the efficacy of the proposed adaptive segment-aware K-push algorithm.

Chapter 6 presents an adaptive segment prefetching strategy for improving the end-user QoE while reducing the client-video source access latency in Multi-access Edge Computing networks. The chapter provides a complete description of the RAN link throughput prediction mechanism based on the Hidden Markov Model and virtual client buffer inference mechanism. Then, the chapter presents a detailed discussion of the experimental results to demonstrate the efficiency of the proposed adaptive segment prefetching strategy in terms of improving video QoE and segment hit ratio.

Finally, **Chapter 7** concludes the thesis with a summarization of the research contributions and suggests the directions for possible future works over the Internet video streaming technology.

CHAPTER

2

Background and Literature Survey

In this chapter, we first describe the general architecture of Dynamic Adaptive Streaming over HTTP (DASH). The DASH is a very broad standard and we just provide a brief overview of some essential features and mechanisms related to our thesis. Next, we describe the quality of experience (QoE) together with the essential details for measuring QoE. We also provide an overview of Dynamic Adaptive Streaming over HTTP/2 and Multi-access Edge Computing (MEC). We then present an elaborated survey of the literature pertaining to bitrate adaptation algorithms for DASH (in Section 2.5.1), QoE estimation models for HTTP adaptive streaming (in Section 2.5.2), and prefetching and caching Strategies for DASH (in Section 2.5.3). The organization of this chapter is depicted in Fig. 2.1.

2.1 Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP, colloquially known as MPEG-DASH, is a vendor-independent international standard finalized in April 2012 [6, 7] and popularly used for HTTP-based adaptive video-on-demand streaming. This new technology used to deliver online video content overcomes several limitations of the traditional progressive streaming, such as wastage of bandwidth due to aggressive buffering and lack of adaptation to bandwidth variation [8]. The Vlaamse Radio- en Televisieomroeporganisatie (VRT), a national public-service broadcaster of Belgium, has performed the first major trial of DASH in 2012 for the London Olympics. The VRT has offered its audience the chance to experience the Olympic

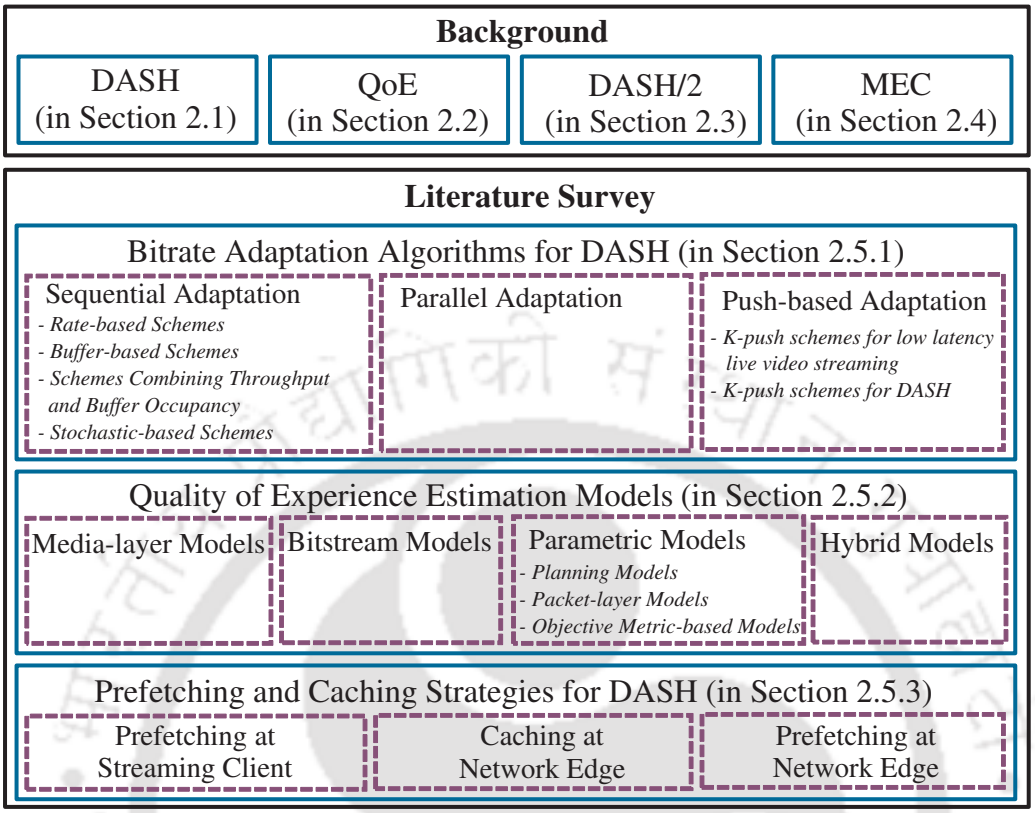


Figure 2.1: Organization of background and literature survey chapter.

Games broadcast on their personal devices via the DASH to demonstrate the benefits of the standard for adaptive streaming. Of late, several leading service providers in the video space, including YouTube, Netflix, and Hotstar, have been the major drivers in DASH's adaption and use DASH to deliver the video content to various devices (including mobile phones). The popularity of DASH for video streaming is mostly accredited to the use of HTTP together with TCP. The usage of HTTP by DASH yields several benefits [8, 41]; a few of them are listed as follows:

1. The HTTP traffic has the ability to traverse firewalls, network address translation (NAT) devices, and middleboxes effortlessly [42]. Thus, the use of standard HTTP protocol for DASH provides the HTTP traffic a more ubiquitous reach to clients.

2. DASH also facilitates the streaming of video content from regular commodity HTTP Web servers [8, 41]. This will not only enable the video streaming service providers to

use the existing and significantly inexpensive streaming infrastructures to maximum advantage but also cuts down significantly on the operational costs. Moreover, the use of existing HTTP infrastructure allows the network service providers to deploy caches to minimize the load on the network and improve the overall network performance.

3. The HTTP-based delivery provides the ability to move control of "streaming session" entirely to the streaming client. In addition, DASH provides the streaming client a simple means to seamlessly adapt the video segment bitrate on-the-fly in reaction to the available bandwidth perceived and user preferences without requiring the negotiation with the streaming server. The client makes each video segment/fragment request independently and maintains the playback session information. Thus, the servers do not require any session state tracking and facilitate the client to fetch the video segments from multiple HTTP servers. DASH clients retrieving the video segments from multiple servers leverage the existing CDNs and proxy caches to provide load-balancing and fault tolerance between HTTP Web servers [43, 44].

4. The use of TCP reliability and inter-flow friendliness ensures each streaming client gets a fair fraction of the network bandwidth for its streaming traffic while sharing the limited network bandwidth with other traffic [41].

Fig. 2.2 illustrates a simple adaptive streaming scenario between an HTTP server and a DASH client. The key concept in DASH is to encode the raw video file into multiple representations/bitrate levels or spatial resolutions. Each representation fundamentally varies in encoding rate, display resolution, and format. Subsequently, each of these encoded video files is split/chopped into multiple *segments/fragments* of a fixed playback duration (typically $1 - 15$ seconds). For each video, a Media Presentation Description (MPD) file is prepared to describe the temporal and structural relationships between the segments. This is often referred to as the encoding and segmentation process. The video segments and the MPD file are hosted on the DASH streaming server (as shown in Fig. 2.2).

Media Presentation Description: In general, the MPD is an XML file. It represents the different bitrate qualities of the video content and the individual segments with unique Uniform Resource Locators (URLs) for each bitrate quality. In other words, the MPD

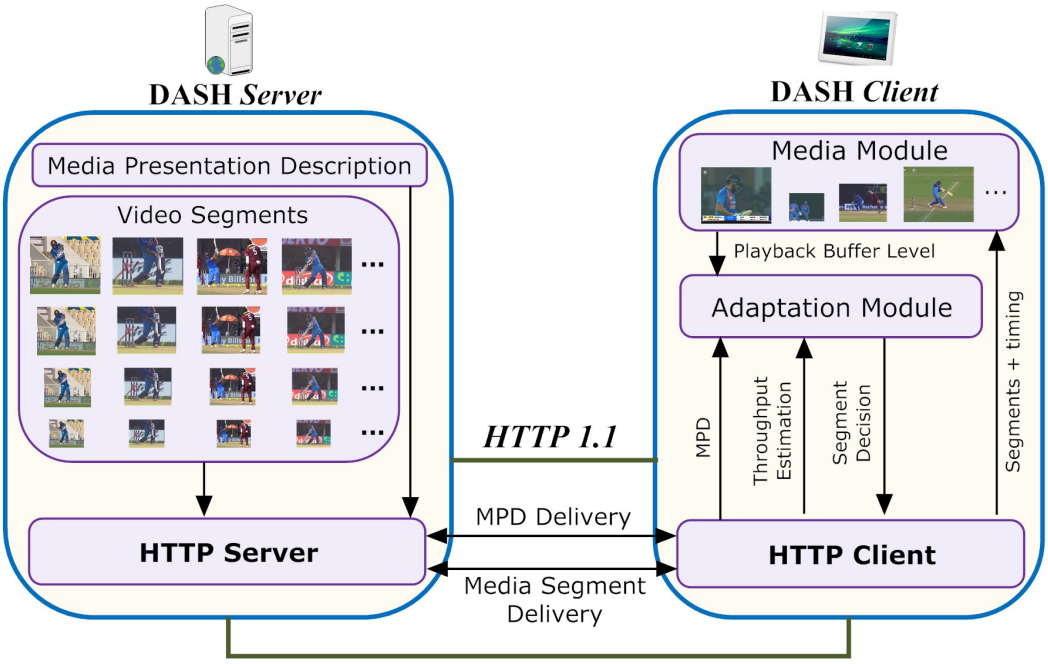


Figure 2.2: Principle of adaptive streaming in a typical DASH system.

structure provides the binding of the segments to the video bitrate, codec used, segment start time, duration of segments, etc. The DASH Media Presentation Description is a hierarchical data model, as shown in Fig. 2.3, where each could contain one or more **Periods**. Next, each of those **Periods** contains several media components such as video components, audio components, and subtitle or caption components. The video components could represent the actual content (e.g., a movie or a football game) with different view angles or with different codecs. The audio components could be used for representing different languages or with different types of information (e.g., with director's comments, etc.). Typically multimedia components such as video, audio or subtitles/captions, etc. are arranged in **AdaptationSets** (as depicted in Fig. 2.3). Thus, each **Period** could contain one or more **AdaptationSets** that enable the grouping of different components (e.g., with the same codec, language, audio channel format, etc.) that logically belong together.

As shown in Fig. 2.3, an **AdaptationSet** consists of a set of **Representations** containing interchangeable versions of the actual content (e.g., different resolutions, bitrates, etc). The **Representations** are divided into **Segments** to enable the switching between different representations during the video playback. These **Segments** are described by a URL and in certain cases by an additional byte range if those segments are stored in a bigger,

continuous file. Note that DASH does not restrict the segment duration or give specific guidelines on the optimal segment duration. The duration of the segments can be chosen depending on the given scenario. For example, segments with longer play duration allow more efficient compression or result in lower network overhead, as each segment request will introduce a certain amount of HTTP overhead. On the contrary, shorter duration segments - widely used for live scenarios - enable faster and flexible switching between individual bitrates under highly variable bandwidth conditions like mobile networks. In addition, DASH does not allow arbitrary switching between the representations at any point in the stream and certain constraints have to be considered during the playback of the content. Thus, the segments are not allowed to overlap (i.e., each segment will have a well-defined start time) and also have no dependencies between segments.

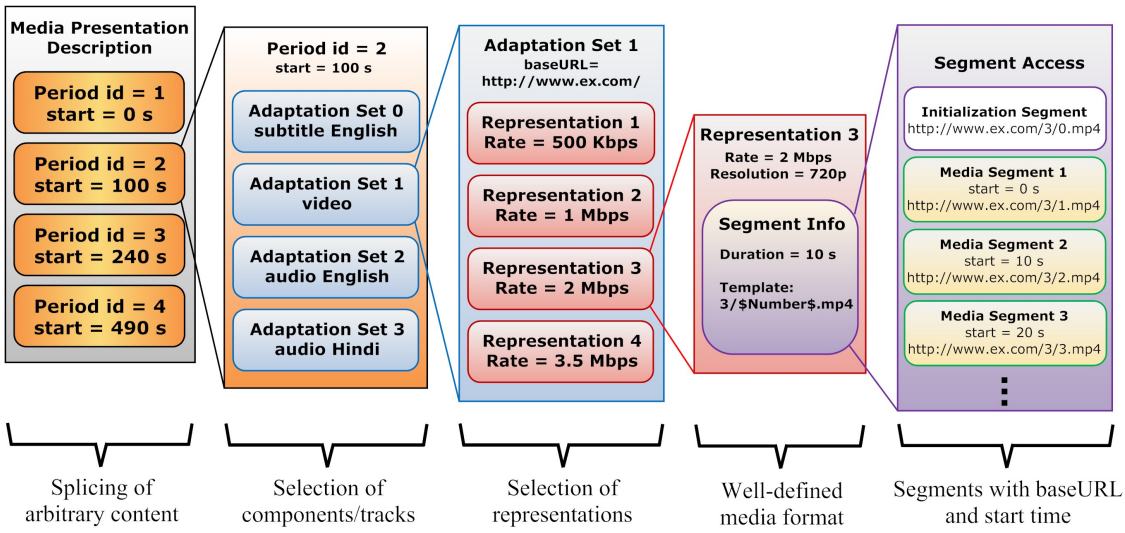


Figure 2.3: Hierarchical DASH data model of the Media Presentation Description, adapted from [1, 2].

The DASH client (i.e., media player) first makes a request and obtains the MPD to play the actual media content. By parsing the MPD, the DASH client learns about all the information required to fetch the desired segments during the video session. This information typically includes various encoded alternatives of the multimedia content, AdaptationSet availability, video resolutions, minimum and maximum bandwidths of the representations/bitrate levels, segment accessibility features and required digital rights management. Typically, the DASH client has two key modules, namely the media module

and the adaptation module (as shown in Fig. 2.2). The media module is responsible for rendering the video frames from the playback buffer. In contrast, the adaptation module is responsible for selecting the suitable representation for the next segment to be downloaded. Once the adaptation module decides the appropriate segment representation, the DASH client starts fetching the segment(s) using HTTP GET requests. The *bitrate adaptation algorithm* employed by the adaptation module could either be conservative or aggressive in changing the representation. In fact, this might take it anywhere between the two extremes: lower bitrate vs. excessive playback interruptions. Hence, this would influence the viewing experience of the end-user [45].

2.2 Quality of Experience

In recent times, the video content mostly streamed using DASH represents a large portion of the Internet traffic [5]. In addition, its relevance is expected to increase in the years to come. This aspect poses a severe challenge on how to deliver the DASH video content over the Internet efficiently. Effective delivery of DASH streams depends on the degree to which the streaming service providers meet the end-users expectations, the so-called Quality of Experience (QoE).

In general, QoE is defined as the overall acceptability of an application or service, as perceived by the end user [10]. Of late, the EU Qualinet community formally defines QoE as: "*QoE 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*" [46, 47]. The traditional Quality of Service (QoS), limited to telecommunication services, purely relies on the technical measurements for measuring the service's efficacy. On the contrary, QoE takes the end-user related factors such as his/her expectations with respect to the service and their level of satisfaction into consideration. Several factors typically influence the overall acceptability of the video streaming service. Several communities and researchers proposed classifications of QoE influence factors into multiple dimensions. For example, the EU Qualinet community [46] groups the QoE influence factors into three categories; (1) system factors, (2) human factors, and (3) context factors. We list examples of a few QoE influencing factors across the categories mentioned earlier, only illustrative and not

exhaustive, in Table 2.1.

Table 2.1: Example of few QoE influencing factors under different categories.

SYSTEM FACTORS	**- Device:** Screen size and resolution, Battery lifetime, CPU, Memory, etc. **- Network:** Network bandwidth, Packet loss rate, Data cost, Delay, etc. **- Server:** Video Codec, Delivery strategy, Segment size, Duration, etc. **- Content:** Encoding rate, Play duration, Video age, etc.
HUMAN FACTORS	User profile, Gender, Age, User expectation, Usage pattern, etc.
CONTEXT FACTORS	**- Physical:** Location, Usage time, Available network, etc. **- Social and Cultural:** Sharing or solitary, Device selection, Content popularity, etc. **- Temporal:** Available time, Willingness, Network switch, etc. **- Task:** Other tasks, Usage requirements, etc.

With the dynamics of end-to-end path bandwidth and an inappropriate adaptation algorithm, the user can have a poor QoE that ultimately leads to a decline in the popularity of the service [9]. In addition, video streaming in resource-constrained mobile devices that operate on dynamically changing and especially low-bandwidth mobile networks (particularly true for many developing countries) poses quite a different challenge for content providers. In the light of this, ensuring a satisfactory viewing experience during the entire video session is very important for user engagement and revenue generation, both for the network and content providers [11, 12]. The QoE plays a vital role in successfully deploying online video streaming services, improving the existing service delivery process, and developing future streaming technologies. Thus, accurately measuring or estimating the QoE is paramount for the service providers.

The ITU-T Recommendation P.10/G.100 Amendment 5 defines: "*QoE assessment as the process of measuring or estimating the QoE for a set of users of an application or a service with a dedicated procedure, and considering the influencing factors*" [47]. In general, the QoE assessment methodologies can be categorized into two categories, namely subjective and objective. For subjective QoE assessment, subjective experiments are carried

out where a participant is asked to watch the test video clip(s) in a controlled laboratory environment. The subjective experiments may be broadly divided into two types: passive and interactive [48].

- **Passive Subjective Experiment:** In a passive subjective experiment, participants are presented with pre-recorded test samples representing the conditions of interest [48]. The participants are asked to passively listen to and/or watch the test material. Then the participant is asked to grade their perception of video quality. Their opinion is collected in the form of wordings used to define the subjective rating scale.

- **Interactive Subjective Experiment:** In an interactive subjective experiment, two or more subjects actively engage in conversation using equipment designed to emulate the use cases of interest [48]. The subjects are often given tasks in order to stimulate conversation and interaction. In general, most subjective experiments tend to be passive in nature. However, there are some aspects of user experience, for example, the effects of delay and echo, which only become apparent in conversational scenarios.

The absolute category rating (ACR) is a single-stimulus subjective test method for passive subjective experiments set forth in ITU telecommunication recommendation [49]. The ACR method is the most acceptable method for subjective QoE assessment using a discrete rating scale independently without using an explicit reference. Mean Opinion Score (MOS) is the most popular and the de facto metric for subjective QoE assessment. The MOS is determined by averaging the individual rating scores collected from a large number of participants. Although MOS provides a better understanding of the end-users viewing experience, the subjective QoE assessment has inherent drawbacks. Subjective QoE assessment incurs costs and time and not practical for real-world applications as obtaining the MOS feedback directly from the viewer is not easy. Furthermore, conducting the large-scale MOS measurements to obtain the subjective perception that is statistically meaningful is an onerous job.

In recent years, as an alternative solution to the subjective QoE assessment approach, researchers have envisaged objective QoE assessment to predict/estimate the overall viewing experience leveraging either the network-related factors or a set of industry-standard objective QoE metrics [21]. These industry-standard objective QoE metrics can be quantified directly

either with an intermediate measurement tool in the network or with a lightweight client-side instrumentation script. The industry-standard objective QoE metrics include but are not limited to average video quality, number of interruptions, duration of interruptions, number of bitrate switching events, and initial startup delay [8]. We provide a summary of a few objective QoE metrics in the literature as follows.

1) *Average Video Quality*: The quality of a video stream is based on the encoding rate. The encoding rate is the average data required to play one second of the video. Unlike the progressive download-based video streams, which stick to the same encoding rate for the entire video session, adaptive bitrate streaming varies the encoding rate for the video segments depending on the network conditions. Thus, streaming segments with a higher encoding rate would usually result in higher quality video and have a distinct effect on the user's QoE. The average video quality metric is defined as the ratio of the sum of individual segment encoding rates and the total number of segments downloaded during the video session.

2) *Number of Interruptions*: In general, the video playback is not interrupted as long as the rate at which the playback buffer is being filled is greater than or equal to the rate at which the video is played. An interruption of video playback occurs if the buffer gets depleted and the media player waits for a certain duration for the buffer to be partially filled before resuming the playback. The frequency with which the playback interruptions occur could significantly impact the end-user viewing experience and leads to poor user engagement. The number of interruptions metrics represents the total number of playback interruptions recorded during the entire video session.

3) *Duration of Interruptions*: The duration of interruptions represents the time taken to fill the playback buffer with video content of the desired bitrate and resume the playback. The users are likely to be dissatisfied in case of interruptions of longer duration and have a distinct effect on the end-user QoE.

4) *Number of Bitrate Switching Events*: In adaptive bitrate streaming, the streaming client tends to select video bitrate for the segments based on the network conditions. The segment bitrate could gradually increase before settling at a suitable video bitrate with favorable network conditions. On the contrary, the segment bitrate could be

reduced due to degraded network conditions or interruptions. The change in the video bitrate can be easily perceivable to the and-user. Thus, it is necessary to ensure that the number of bitrate switching events is reduced.

5) *Initial Startup Delay*: The initial startup delay typically includes the time taken to load the HTML page (or MPD file), load the media player plug-in, and build the initial playback buffer reservoir. A longer initial startup delay, i.e., more than 2 seconds, could result in the viewer abandoning the video completely [8]. Hence, the initial startup delay is an essential factor that affects the end-user QoE.

Nonetheless, these industry-standard objective QoE metrics have the issue of inter-dependency. They are interrelated in complex, counter-intuitive ways, and their relationship can be unpredictable. The underlying network bandwidth variations and video content characteristics can further complicate the interactions as well. To address the issue of inter-dependency and develop a unified and quantitative understanding of these metrics' impact on QoE, several QoE prediction/estimation models have been developed. These QoE prediction/estimation models correctly determine a quality score using the aforementioned network-related factors or objective QoE metrics. In practice, the QoE estimation models, especially for mobile video streaming, can be used by a variety of stakeholders like mobile network operators, video streaming service/platform providers, and regulators. We provide a discussion of the QoE estimation models in Section 2.5.2

2.3 Dynamic Adaptive Streaming over HTTP/2

In recent years, DASH over HTTP/1.1 [50] has been the widely used streaming technologies by several popular video content providers, such as YouTube and Netflix, mostly to deliver their videos specifically to mobile devices. Nonetheless, DASH over HTTP/1.1 [50] demands an accurate bandwidth prediction for providing the best viewing experience to end-user. In addition, the design limitations of HTTP/1.1, namely head-of-line blocking, fat request/response headers, and client-initiated requests, impact not only Internet web traffic delivery but also Internet content delivery in general.

In DASH, segments of shorter play duration benefit the streaming to fully utilize the available bandwidth. However, HTTP/1.1 essentially imposes some additional overhead in

terms of request messages for fetching the video segments. This is also known as *request explosion* problem [23]. The request message overhead will be significantly high when the streaming session uses segments of a smaller play duration. Figure 2.4 illustrates the number of resource requests generated for various segment play duration (typically 1-15 seconds) by the client media player while streaming the *Big Buck Bunny* video sequence with a total play duration of 596 seconds. The request message overhead mostly leads to increased delay as it takes more than one round-trip time (RTT) before the next segment is downloaded. The overall response latency could increase greatly, especially in mobile networks, where the request messages could pile up due to constrained uplink bandwidth. Hence, it would influence the end-user viewing experience with highly dynamic network conditions and can also have degradation in the quality of experience.

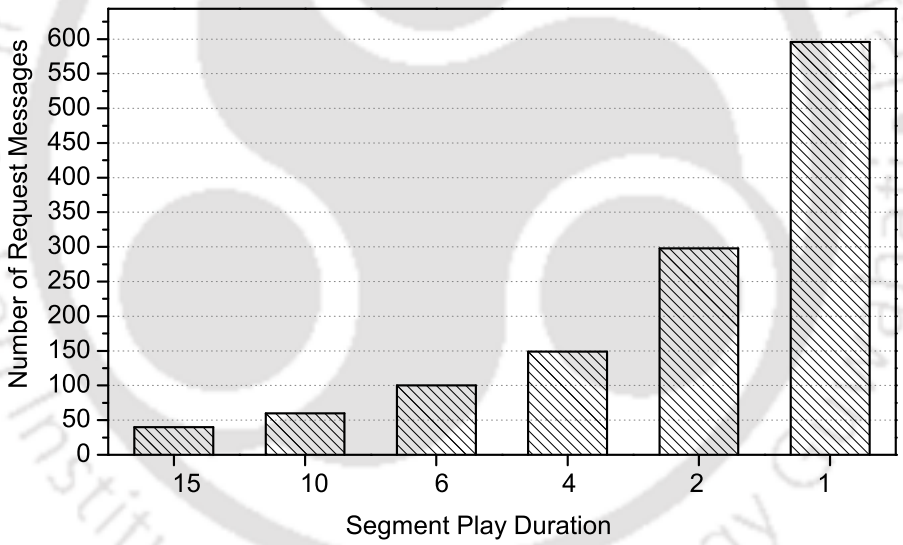


Figure 2.4: Number of segment requests generated for various segment play duration while streaming Big Buck Bunny video sequence.

To address the design limitations of HTTP/1.1, Hypertext Transfer Protocol Version 2 (HTTP/2) [51] (subsequently originated from SPDY/3 draft) has been standardized as IETF RFC 7540 in February 2015 and promises significant performance improvements over its predecessor. The HTTP/2 has started to take the place of and expected to eventually fully supersede its predecessor HTTP/1.1 for web access, mainly due to its improved

performance. The use of fewer TCP connections by the HTTP/2 protocol as compared to the HTTP/1.1 makes it more friendly to the network. This results in less competition with other flows and longer-lived connections, leading to efficient utilization of the available network resources. The HTTP/2 not only retains the same semantics as HTTP/1.1 (includes HTTP methods and status codes) but also adds several key features. We provide a brief discussion of two prominent HTTP/2 features related to the thesis.

1. **Server Push:** This is a powerful new feature introduced in HTTP/2. This feature is also known as "cache push". This feature allows the server to push multiple responses for a single client request. In addition to the primary response, the server also pushes additional associated resources to the user before the user has even asked for them, with an underlying assumption that the client will need the resource(s) in the near future. In general, the HTTP/2 server push is based on the client's good faith of accepting a promise sent by a server for the resource(s) the client will need soon. In all HTTP/2 servers, the push streams are initiated by sending a PUSH_PROMISE frame (same as an HTTP response) on the existing client-initiated stream. The PUSH_PROMISE frame has two purposes: (i) contains the header information of the resource to be pushed, and (ii) notifies the client about the new stream ID for pushing the promised resource. This push order is important because it notifies the client of which resources the server intends to send prior. There are two main approaches[1]for implementing server push: (i) hinted push and (ii) rule-based proxy push.

2. **Stream Termination:** Once the client receives a PUSH_PROMISE frame, it has the option to accept or decline the stream. In particular, if the client already has the unexpired resource in its cache, then pushing it is wasteful. In general, pushing unnecessary contents is undesired in cellular networks where customers are charged by the amount of transferred bytes. The client can completely opt-out or swiftly terminate a pushed stream in between with a RST_STREAM frame.

The performance limitations observed in DASH over HTTP/1.1 can be addressed using the Dynamic Adaptive Streaming over HTTP/2 (DASH/2). DASH/2 provides unprecedented opportunities to eliminate the *request explosion* problem and further improve the DASH

[1]https://blogs.akamai.com/2017/03/http2-server-push-the-what-how-and-why.html

performance by leveraging the HTTP/2 *Server Push* feature, especially for mobile clients. In DASH/2, the server push feature is usually applied to push associated resources/segments prior to receiving the client's request. This effectively reduces the latency in serving the video content to the client. The number of request messages generated at the client-side for downloading the video segments can also be greatly reduced using the HTTP/2 server push feature. On the contrary, the stream termination feature is an effective solution to cancel any undesired transmission from the streaming server. This still has an overhead of about half of an RTT when a stream termination is requested from the client. This feature can terminate the undesired segment transfer in the push cycle if any network fluctuations are observed.

Recently, leveraging the HTTP/2 server-initiated push mechanism for DASH segment delivery has attracted several research efforts. A few pioneering studies have designed K-push schemes for both low latency live streaming [23, 24, 25] and DASH-compliant video streaming [26, 27]. We provide a brief discussion of the K-push schemes in the literature in Section 2.5.1.

2.4 Multi-access Edge Computing

In the past two decades, mobile cellular networks have experienced four generations with tremendous advancements in both wireless communications and information technology. Simultaneously, several new requirements in end-users' devices are appearing and various constraints are raised in terms of battery lifetime, computational power, and memory limitations due to the advent of new kinds of mobile applications, such as HTTP adaptive streaming. With the rise of such mobile applications, the end-users' demands from mobile cellular networks also get more strict, including assurance of ultra-high data rate and extremely lower access latency. Thus, the traditional base station-centric network architecture becomes inadequate for fulfilling these user requirements. Moreover, efficient network resources management (due to data storage and processing) in order to enable the new applications/services has been a significant challenge for Mobile Network Operators (MNOs).

In tandem, adaptive bitrate streaming solutions, especially DASH with segment-based flexibility, also bring a potential benefit for the end user's QoE. Currently, DASH is widely deployed on the Internet and accelerated by several leading video content providers such as

YouTube, Netflix, and Vimeo for delivering video(s) to mobile users. This also contributes to a considerable proportion of the data traffic generated on the Internet. According to Cisco's recent global mobile data traffic forecast, mobile video streaming will account for nearly four-fifths (79 percent) proportion of the world's mobile data traffic by 2022 [5]. Meanwhile, such growth rate and bandwidth demand create immense pressure on mobile network operators. In addition, the inevitable bandwidth jitter governed by the backhaul link congestion and server processing delays could lead to frequent bitrate switching events and even unnecessary playback interruptions in DASH video delivery. With the surge in the number of mobile users requesting video content over the Internet, the video service providers also started observing complications related to their services' performance and scalability. Hence, DASH video playback sessions with frequent video bitrate oscillation and interruptions could severely degrade the end-user QoE and translate into a decline in popularity for the video service providers.

In the wake of these problems, with an ultimate goal to provide the expected level of QoE to end-users, the video service providers moved beyond traditional client-server streaming and embraced content delivery networks (CDNs) for improving their video content distribution. The video content pre-cached at the CDN servers (located near the network edge) significantly reduces the video segment access latency as requests to the same content from mobile users can be accommodated easily without duplicate transmissions from source video servers. Thus, the end-users are likely to get improved video QoE. Nevertheless, there exist several limitations that inhibit the video service providers from using the CDN approach for the delivery of multimedia content [29, 30]. Some of these problems are summarized below.

1. **Lack of flexibility:** In general, business contracts with third-party CDN providers are very expensive. They also typically involve prior investments for reserving/possessing adequate capabilities and resources. The multimedia service providers have to make necessary payments for the reserved resources even when they are not used optimally. In addition, the multimedia service provider can not flexibly decide and deploy a selected subset of segments for the video (e.g., popular videos) at the CDN server(s) without incurring extra charges from the CDN service provider.

2. **Lack of control and limited access:** The video quality of experience for the

end-users while accessing the multimedia content can vary greatly depending on the CDN surrogate servers' performance. The multimedia service providers encounter trouble in monitoring the global performance of CDN surrogate servers and assuring service availability. During such times, the multimedia service providers have to depend entirely on the CDN operators' performance feedback. Often, these problems are discovered only after the end-users have an unpleasant viewing experience. The degraded viewing experience could often result in a decline in the service popularity. Besides, the video content manipulation policies (e.g., caching or prefetching) at the CDN surrogate servers are entirely administered by CDN operators. Thus, video service providers have no control over these content manipulation policies. The video service provider also can not dynamically adjust the content manipulation policies based on the real-time context of the networks and/or the end-users.

3. **Limited Points of Presence (PoP):** Typically, the CDN operators, especially third-party CDNs, deploy the distributed infrastructure in strategic locations in order to cope with the ever-increasing user demands. Nevertheless, during flash crowd (i.e., an unusual surge in the number of user requests from a particular location) with CDN PoPs located far away from the source of demand may be problematic for distributing video content. Such scenarios could lead to an increase in access latency for the video segments. Thus, it could also significantly degrades the end-users viewing experience and might disrupt the service.

To address the challenges as mentioned earlier, *Mobile Edge Computing* was introduced as a promising approach by the standards organization European Telecommunications Standards Institute (ETSI) in December 2014 [31]. In September 2016, the ETSI Industry Specification Group (ISG) had dropped the 'Mobile' out of *Mobile Edge Computing* and renamed it as *Multi-access Edge Computing* (MEC) in order to broaden its applicability into heterogeneous networks [52]. The original definition of MEC in [53] refers to it as a new platform (a.k.a. MEC server) that "provides IT and cloud computing capabilities within the Radio Access Network (RAN) in close proximity to the mobile subscribers". The MEC is envisioned as a virtualized platform that leverages the recent advancements in network function virtualization (NFV), which enables the center of gravity (i.e., network functions, resources, and contents) to move from the network core to the network edge. The

network resources mainly include computing power, storage, and communication resources. In particular, the NFV enables a single edge device to provide computing services to multiple end-user devices by creating multiple virtual machines (VMs[2]). Meanwhile, the VMs can be reused by the applications simultaneously for performing computation-intensive tasks or providing various network functions, which is beneficial for the MNOs [31].

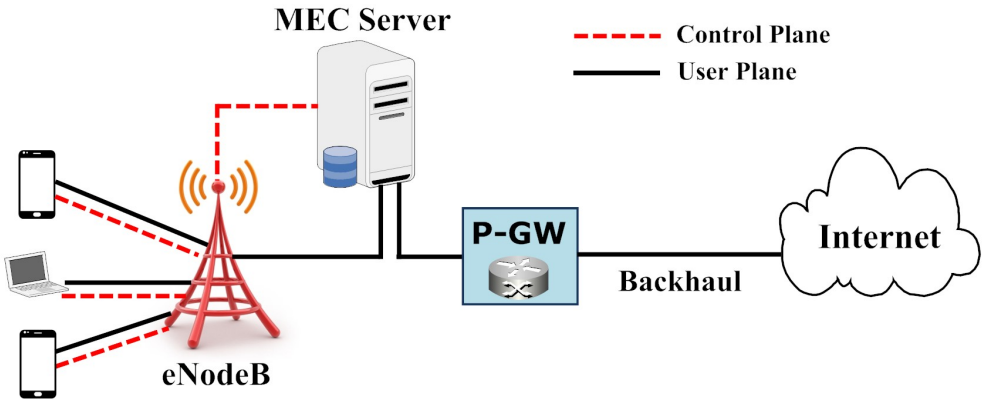


Figure 2.5: An overview of Multi-access Edge Computing architecture [3].

The objective of ETSI Industry Specification Group (ISG) on MEC is to deliver a standard MEC architecture and industry-standardized application programming interfaces (APIs), which can be widely accessible by content/service providers and third party innovative applications [55]. Accordingly, the ETSI ISG does not deliberately document a specific deployment location for the MEC server, leaving its deployment approaches to continued innovation. Of late, the MEC research has drawn much attention of industries and academia, and several researchers have investigated different MEC server deployment approaches at the network edge [3]. The MEC servers located in the proximity of the LTE macro base station (also called eNodeB) are well-suited in a standard MEC environment, mostly based on the technical parameters that include access latency, resource requirement, scalability, availability, and deployment cost. We present a high-level overview of MEC architecture with the MEC server located in close proximity of base stations (i.e., close to eNodeBs) in Fig. 2.5. The MEC server in Fig. 2.5 can either efficiently handle an end-user request(s) and respond to the request(s) directly or forward the user request(s) to source

[2]The VM is a virtual computer mapped to the physical machine's dedicated hardware using software technologies, which can provide the capability for virtual computing (i.e., CPU, memory, hard drive, etc.) and network interface to heterogeneous client devices [54].

servers located on the public Internet.

2.5 Literature Survey

In the last decade, there has been a tremendous interest in multimedia streaming, especially for HTTP adaptive streaming from both media-processing and networking research communities. In the literature, various aspects of dynamic adaptive streaming over HTTP are explored to improve the end-users QoE in mobile devices. In this section, the progress in the field of improving QoE for DASH has been discussed with critical analysis, and several design issues are pointed out that are partially addressed or remained unsolved till now. For ease of presentation, the complete discussion has been classified into three groups - bitrate adaptation algorithms for DASH, QoE estimation models, and prefetching and caching strategies for DASH. In what follows, we present a brief summary of the works addressing each of these aspects.

2.5.1 Bitrate Adaptation Algorithms for DASH

In recent times, DASH has emerged as a dominant technology standard for video delivery over the Internet. It has attracted remarkably great attention from researchers, despite being a comparatively recent video-on-demand streaming technology. Popular online video-on-demand service providers such as Netflix, Hulu, and Hotstar, use DASH [56]. The MPEG-DASH standard purposely does not designate a specific adaptive bitrate (ABR) streaming logic/algorithm. Thus, it leaves the design of an adaptive bitrate streaming algorithm to continued innovation. We first provide a high-level classification of the bitrate adaptation algorithms for DASH as shown in Fig. 2.6. The high-level classification in Fig. 2.6 is primarily based on the bitrate adaptation logics deployment location, video segment request and download pattern from the streaming server, and parameters they consider for making an informed segment bitrate decision. To begin with, the bitrate adaptation algorithms based on the location where the adaptation logic is employed can be broadly classified into four categories; server-side, network-assisted, client-side, and hybrid [13]. We begin with a brief summary of the works undertaken in each of the categories mentioned above, along with their limitations in the rest of the section.

The server-side adaptation solutions [57, 58] implicitly control the bitrate streamed by

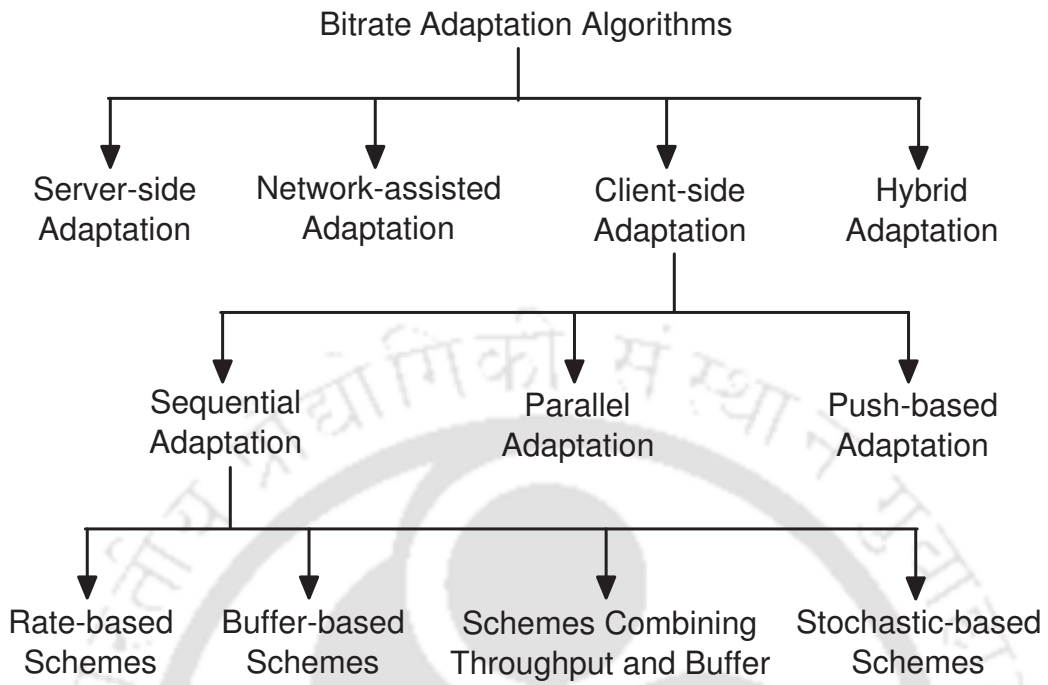


Figure 2.6: Overview of bitrate adaptation algorithms classification for DASH.

a server-side shaping method without any assistance from the client. These solutions aim to maintain a stable playback buffer at the client and do not take user-perceived QoE into consideration. Bruneau-Queyreix et al. [59] proposed a multiple-source adaptive streaming solution, wherein the video segment is further divided into subsegments and hosted across multiple streaming servers to improve the QoE. The server-side adaptation solutions not only incur higher overhead when the number of clients increases (as they maintain the session information for each client) but also require modifications to the traditional HTTP server.

Several network-assisted solutions [60, 61, 62, 63, 64] have been proposed to make an informed in-network bitrate adaptation. The work in [60] proposed a video control plane to enable the network elements running a centralized algorithm to leverage the video streaming flow information and compute the video bitrate. Bouten et al. [62] proposed a QoE-driven network-assisted optimization system (using a set of in-network agents acting as proxies) to address the problem of multiple clients competing for the shared network link. Similarly, Petrangeli et al. [63] proposed a QoE-driven network-assisted adaptation algorithm to address the fairness issues with multiple clients. In their approach, the in-network components

inform the current network condition to the client(s) for making the best possible bitrate decision. However, these approaches have a high overhead of communication with the clients and in-network agents' failure could result in degraded QoE.

The hybrid approach consists of both server-and-network-assisted DASH (SAND) and client-and-network-assisted DASH (CAND) architectures. For instance, software-defined networking (SDN) can be used in concert with either the client or the server-side for a hybrid mechanism. An example of a SAND approach with SDN is [65] while an example of a CAND approach with SDN is [66].

In tandem, several pioneering studies choose to take up the viewpoint of a streaming client for bitrate adaptation while assuming the server/network environment entirely to be a black box. The client-side rate adaptation strategies (such as the ones in [45, 56, 39, 37, 67, 68, 69, 70] and the references therein) have an advantage over both the server-side or network-assisted adaptation. These advantages include the flexibility to deploy the solutions straightaway without any modifications to the server implementation. Furthermore, these approaches have the ability to detect and respond to network bandwidth fluctuations promptly. Based on the client's segment request and download pattern from the streaming server(s), the existing client-side bitrate adaptation strategies in the literature can be broadly classified into three main classes, namely: sequential adaptation, parallel adaptation, and push-based adaptation. In what follows, we present a discussion of the related works in client-side bitrate adaptation strategies using this classification in the rest of the subsection.

Sequential Adaptation

In sequential adaptation, the video segments are downloaded in sequential order, i.e., one after the other, where the next video segment cannot be requested until the previously requested segment is fully downloaded. The sequential adaptation schemes can be predominantly classified into four broad categories, based on the heuristic of the ABR controller: rate-based, buffer-based, combining throughput and buffer, and stochastic-based. These categories of algorithms differ from each other not only in their basic approach of estimating the available bandwidth but also in terms of the parameters they optimize. The sequential bitrate adaptation algorithms with the nuances and technicalities in the literature have been comprehensively discussed in [8, 13, 41]. In the rest of this subsection, we overview

existing algorithms in the literature based on their category and highlight their features relevant to our thesis.

Rate-based Schemes: The rate-based schemes adapt the video bitrate on-the-fly based on the perceived network bandwidth with an aim to match the selected bitrate to the end-to-end estimated bandwidth. One of the earliest implementations of DASH was in the Adobe Open Source Media Framework (OSMF) player[3]. The adaptation algorithm used by the OSMF rate adaptation module purely relied on historical network throughput. The historical network throughput is usually estimated by measuring the time taken to download previous video segments, which in turn is used to determine the appropriate video bitrate for the next segment to be fetched. According to the analysis in [15], the adaptation algorithm employed in OSMF fails to converge to a single bitrate but keeps oscillating between the lowest and highest bitrate levels. Using a similar approach, the algorithm in [71] used the ratio of the average time to fetch a segment over the time taken to fetch the last segment to decide the bandwidth. The estimated bandwidth is then used to determine the appropriate bitrate level for the next segment to be fetched. The work also proposed a gradual bitrate switching strategy instead of switching abruptly. Thus, this approach makes the adaptation more conservative while switching to higher bitrates and aggressive while heading downhill. However, this QoE-aware DASH system requires an intermediate measurement proxy and additional hardware. The proxy-based approach also limits the scalability of the solution. The authors in [72] proposed a bitrate adaptation algorithm that takes into account the bitrate of forthcoming segments and estimated bandwidth while choosing representation for the next video segment.

The authors in [73] have demonstrated that the off-intervals in HTTP Adaptive Streaming (HAS) become a source of ambiguity and prevent the video client from perceiving the available network bandwidth correctly. Thus, it also prevents the client from making accurate rate adaptation decisions for future video segments. To address this limitation, the work in [73] proposed a client-side bitrate adaptation algorithm based on a "probe and adapt" principle and presents the same as a present a four-step model. In the presence of off-intervals, the adaptation algorithm estimates the network bandwidth using a proactive probing mechanism with an end goal to minimize video bitrate oscillations. The probing

[3]http://blogs.adobe.com/osmf/

mechanism constantly probes by incrementing its sending rate and prepares to back off once it experiences congestion. In the scheduling step, the scheduler drives the buffer level towards a predefined maximum buffer occupancy level. In the meantime, the algorithm also determines the target inter-request time based on the smoothed value of the available bandwidth (estimated from the probes) to dispatch the subsequent segment request.

To address the challenge of unfair bandwidth estimation induced by the off-intervals during the segment downloading process, Liu et al. have designed a probe-based bandwidth estimation method [74]. Their method includes a logarithmic law-based increase probing scheme and a conservative back-off-based decrease probing scheme. The probe-based bandwidth estimation method aims to quickly converge the probed bandwidth to the fair-share bandwidth where multiple clients are competing for the bottleneck network resources. The authors in [74] then designed a fairness-aware smooth rate adaptation (FSRA) approach considering both the playback buffer occupancy and the probed bandwidth. The FSRA approach is based on a dual-threshold. The two thresholds are used as operation points to select an appropriate segment bitrate and obtain a continuous video playback.

Buffer-based Schemes: The buffer-based schemes decide the appropriate video bitrate for the segment(s) merely based on the playback buffer occupancy. The main strength of buffer-based schemes comes from the fact that it provides the best possible bitrate for the segment to maintain a continuous video stream. In addition, it also avoids frequent playback buffer underflows resulting in unnecessary playback interruptions since the target is to ensure the playback buffer does not go empty.

The work in [75] demonstrated that choosing the best representation level for a video segment is difficult since most of the rate adaptation algorithms estimate the available bandwidth using HTTP over TCP. In [76, 77], the authors reiterated the same and presented a pure buffer-based adaptation algorithm that did not use bandwidth estimation of any kind for segment bitrate adaptation. This way, they avoid being too conservative (underestimating the bandwidth and choosing a lower bitrate) or too aggressive (overestimating the bandwidth resulting in frequent interruptions leading to shoddy user experience). Their buffer-based algorithm exclusively considers the current playback buffer occupancy (in playback duration) as a marker for bitrate adaptation. If the buffer level is approaching higher values, then the bitrate for the subsequent segment is upgraded; otherwise, the bitrate for the next

segment is downgraded. The job mentioned above is accomplished by employing a rate map, a monotonically increasing function mapping buffer occupancy with the set of available representations. Usually, an available representation or the one closest to the discrete bitrate suggested by the rate map is chosen for fetching the next segment. The sort of rate map used with the bitrate adaptation algorithm determines the convergence time and responsiveness to underlaying network bandwidth fluctuations.

Spiteri et al. in [37] proposed an online bitrate adaptation algorithm called Buffer Occupancy based Lyapunov Algorithm (BOLA) that uses Lyapunov optimization. The optimal algorithm within the BOLA framework requires only knowledge about the playback buffer occupancy (measured as the number of chunks) and does no requires an estimate of the available bandwidth. The Lyapunov optimization in BOLA is designed to maximize a joint utility function by rewarding an increase in the average video quality and penalizing frequent playback interruptions. Furthermore, a variation of the BOLA called BOLA-O mitigates the severe bitrate oscillations by introducing a form of bitrate capping when switching to higher bitrates.

The work in [78] proposed a buffer-based bitrate adaptation algorithm with dynamic buffer thresholds which focus on continuous playback if the bandwidth fluctuates dramatically and ensures a smooth quality playback under stable network condition. Their approach makes use of dual-threshold to decide the appropriate segment bitrate while dynamically adjust the underflow-threshold to archive the trade-off between smoothness and continuity. The authors in [69] proposed a QoE-aware buffer-based bitrate adaptation scheme for DASH by merely exploiting client-side playback buffer state information to optimize user QoE. The buffer-based scheme adaptively decides an appropriate bitrate for segment(s) using a dynamic buffer-based controller, which leverages the latest information of playback buffer occupancy to stabilize the streaming client buffer level.

Beben et al. in [40] proposed an adaptation & buffer management algorithm, called ABMA+, that follows the concept of bitrate adaptation based on predicted video playout rebuffering probability. The ABMA+ approach continuously estimates segment download time characteristics and exploits a pre-computed playout buffer map to satisfy the assumed rebuffering threshold. The pre-computed buffer map used in ABMA+ determines the size of the playback buffer, which in turn predicts the rebuffering probabilities for the available

video representations. Subsequently, it selects a maximum video representation for fetching the next segment, which guarantees a lower predicted probability of video rebuffering.

Schemes Combining Throughput and Buffer Occupancy: The bitrate adaptation schemes in this category leverage both current playback buffer occupancy and estimated throughput for making an informed bitrate decision for the next video segment.

Miller et al. in [79] proposed a bitrate adaptation algorithm that uses estimated network throughput along with current buffer occupancy (in playback seconds) to decide a suitable video bitrate for the next segment. The main focus of their algorithm remains on avoiding interruptions in playback due to buffer underflows. The algorithm basically uses a combination of average segment throughput over a time interval and the throughput determined while fetching the last segment. In addition, it also maintains three buffer-level thresholds: B_{min}, B_{low} and B_{high} (in seconds of playback time). In case the current playback buffer level ($B(t)$) falls below B_{min}, representation with the lowest bitrate from the available set of representations is chosen for the next segment. If $B(t)$ lies between B_{min} and B_{low}, next lower bitrate level is chosen as appropriate. On the contrary, if the $B(t)$ value is greater than B_{high} and the next higher bitrate level (w. r. t. the bitrate level of the last fetched segment) is greater than the average throughput during a time interval by a certain percentage, then the next higher representation is chosen as appropriate for the next segment. To improve the robustness of the algorithm due to measurement uncertainties, when the buffer level $B(t)$ lies between B_{low} and B_{high}, rather than increasing the bitrate immediately, downloading the subsequent segment is delayed until $B(t)$ falls back to $0.5(B_{high} + B_{low})$. Besides, delaying the next segment fetch instead of switching to higher representation also avoids frequent bitrate switches. The initial segments are fetched with the lowest possible bitrate to minimize the playback startup delay. This is followed by the *fast start* phase that aggressively ramps up the bitrates of the succeeding segments based on the throughput measurements. By tracking the buffer level, the buffer underflows are avoided to a large extent.

The authors in [80] proposed an advanced rate adaptation (ARA) algorithm, as an improvement of KM algorithm [79], which targets fast reaction time at stable network conditions and stabile adaptation regardless of underlying network conditions. The ARA algorithm leverages four scaling factors with an objective to provide the fastest response time

without jeopardizing the stability. The ARA algorithm provides two runtime modes, namely fast startup mode and normal mode. The fast startup mode enables fast reaction time at the beginning of the video session while switching to a higher bitrate level is performed in a more liberal way. It enters the normal mode once sufficient buffer level (i.e., a minimum of five video segments is reached) and employees two rate adaptation functions: non-stepwise and stepwise. The ARA algorithm's non-stepwise function enables faster utilization of favorable network conditions and increases the video bitrate quality. Once the predefined optimal buffer level is achieved, the stepwise function is used to maintain playback smoothness and to avoid frequent bitrate switching.

The term smoothness in bitrate adaptation refers to ignoring temporary fluctuations in network bandwidth and converging to an average bitrate. On the contrary, agility refers to the aggressive response to available network bandwidth changes. Naturally, they do not go hand-in-hand with each other. The authors in [81] proposed Proportional-Integral-Derivative (PID) control-based algorithm, which aims to achieve a trade-off between smoothness and agility in bitrate adaptation. They used both estimated throughput and buffer occupancy for deciding an appropriate representation level for the segment to be fetched. The length of the video playback buffer is provided as a feedback signal to adjust the bitrate adaptation. In order to address the inaccuracy in TCP throughput estimation, the PID controller was used to regulate the segment bitrate. The feedback mechanism of the PID controller computes three errors (i.e., deviation from desired playback buffer level), namely cumulative present error (proportional part (K_P)), accumulation of past error (integral part (K_I)) and prediction of future error (derivative part). The estimated TCP throughput is paired with control coefficients K_P and K_I to determine the target video bitrate for the next segment.

In [81], an improvised version of the PID control-based algorithm was also proposed. Periodic TCP throughput variations can trigger frequent bitrate switching events during bitrate adaptation. Since continuous bitrate changes substantially impact the user viewing experience, an adjustment factor was used while deciding the target bitrate for the next segment. The adjustment factor is the product of three factors: the target buffer size, the trends in buffer growth and current video bitrate. The bitrate switching logic also includes a specific mechanism to enable the trade-off between smoothness and agility. The bitrate

switch will only happen if the target bitrate suggested by the controller is satisfying certain conditions with respect to previously chosen bitrates.

The work in [39] proposed a segment-aware rate adaptation (SARA) algorithm which considers variation in the segment sizes while measuring the throughput. The variation in segment sizes and playback buffer length are considered along with the estimated throughput to predict the time required to download a segment. The media presentation description (MPD) file was enhanced to list the size of each segment. The SARA adaptation module works in four different stages: fast start, additive increase, aggressive switching, and delayed download, depending on the current playback buffer length and pre-configured buffer thresholds. The throughput estimation module was modified to use the weighted harmonic mean of the segment download times. With favourable network conditions, the playback buffer size can go beyond a preset maximum buffer threshold. Under those circumstances, the SARA algorithm delays the segment fetch until the buffer occupancy falls below the maximum buffer threshold level to avoid superfluous use of both network bandwidth and system resources in case the user quits the video early.

The authors in [82] proposed cost-aware buffer management (CBM), an intelligent buffer management strategy with an end goal of minimizing cost induced by unconsumed video data while enduring the end-user viewing experience requirements. The proposed CBM strategy considers the Lyapunov optimization theory to derive the corresponding strategy for cost minimization and to provide a performance guarantee (in terms of freezing, mean freezing time, power-saving utility requirement, etc.) with explicit bounds. The work in [83] proposed a Throughput-Friendly DASH (termed TFDASH) adaptation scheme to well balance the trade-offs among efficiency, stability, and fairness for multiple streaming clients sharing a common link. The TFDASH guarantees both fairness and efficiency (i.e., higher bandwidth utilization) by avoiding the off intervals during the segment downloading process while solves the instability problem with a dual-threshold buffer model. The authors in [84] formulated the bitrate adaptation for DASH as an optimization problem of the multi-stage decision process. In each stage, they have obtained the optimal local solution discretely by evaluating the reward of each action determined by the QoE evaluation model [85].

In [86], the problem of segment delivery for DASH is formulated as a non-convex optimization problem which is constrained by the estimated network bandwidth, segment

download deadlines, available video bitrates, and playback buffer occupancy. Accordingly, the authors have developed a low-complexity algorithm called FastScan [86] to solve the non-convex optimization problem with the objective of maximizing the segment bitrate and ensuring minimum interruption duration. The FastScan algorithm is re-run at the end of every segment download in order to reconsider and decide the bitrate of the next window of segments.

The authors in [87] proposed a bitrate adaptation algorithm called QoE-enhanced adaptation algorithm over DASH (QAAD) to avoid interruptions and minimize video bitrate switching amplitude during the video playback. The bandwidth estimation scheme in QAAD employees a fixed estimation interval period, which is larger than RTT in general and smaller than the segment play duration. In QAAD, the bitrate selection scheme considers the current buffer occupancy and the estimated network bandwidth together to achieve the QoE-enhancement by minimizing the switching amplitude and preserving minimum buffer occupancy during the video playback.

Stochastic-based Schemes: In recent times, several researchers have modeled the video bitrate adaptation as a stochastic optimal control problem. These schemes employed mathematical tools like the Markov Decision Process (MDP), model predictive control (MPC), and Hidden Markov Model (HMM) to model video bitrate adaptation in DASH systems.

Xiang et al. in [88] modeled the bitrate adaptation problem as a finite MDP with an aim to find an optimal streaming strategy in terms of average video quality, interruptions, and playback smoothness. Their proposed approach estimates the bandwidth using a transition probability matrix and makes use of dynamic programming to obtain the optimal MDP solution. The optimal MDP solution is a table where each row consists of a state and corresponding optimal action to improve the end-user perceived QoE. Subsequently, a table look-up is quickly performed to identify the necessary action during the online decision-making process. However, their proposed bitrate adaptation algorithm has only support for streaming scalable video (H.264/SVC). Besides, if the model is applied online with unpredictable network conditions, it may result in erroneous bandwidth estimations leading to degrades QoE. Zhou et al. [67] put forward an MDP-based DASH bitrate adaptation in order to maximize the end-user QoE. They proposed a sub-optimal greedy algorithm to solve the MDP-based optimization problem involving a transition probability of five system

parameters. A major issue is that the number of states for the MDP model is considerably large, which increases both storage and computational requirements. They have evaluated the proposed approach's efficacy by conducted subjective and objective experiments only with desktop users, without any background traffic.

Yin et al. [68] proposed an MPC-based algorithm to optimally combine both predicted throughput and buffer-based feedback information for optimal video bitrate adaptation. Their MPC approach entails predicting the expected network throughput for the next few video segments. Subsequently, it leverages the predicted network throughput to make the optimal bitrate decisions with an end goal of maximizing the QoE. Nevertheless, executing an MPC-based algorithm is challenging as it requires solving a non-trivial discrete optimization problem at each time step. In practice, this imposes additional computational overhead and requires additional software in the form of a solver logic with every media player. In the light of these challenges, the authors in [68] have proposed a mechanism called FastMPC. The FastMPC mechanism follows a table enumeration approach and solves the specific instances of the problem state-space optimally offline to achieve near-optimal performance. The optimal control decisions obtained from the offline computation are then stored and considered for future online use. The authors in [89] developed a cross-session stateful predictor (CS2P) approach based on the data-driven insights gained from a large-scale analysis of throughput stability and predictability. Their approach uses a data aggregator (called prediction engine) to predict throughput by using observed throughputs from past video sessions. In addition, CS2P improves midstream prediction using HMM for each cluster to model the stateful evolution of intra-session throughput. The CS2P approach can be plugged into an adaptation algorithm to execute the actual adaptation decisions. They have demonstrated the proposed approach efficiency in terms of prediction accuracy and QoE by combining with the MPC-based algorithm [68].

In order to deal with multiple streaming clients competing for the available bandwidth in a shared network, QUEuing Theory-based Rate Adaptation (QUETRA) [90] has been developed as a queuing theory based bitrate adaptation. The QUETRA selects an appropriate segment bitrate such that the buffer occupancy stays as close to the buffer slack (calculated based on the current download throughput and a given buffer capacity) as possible. The work in [36] have proposed a system called Pensieve that generates bitrate adaptation

algorithms on the fly using reinforcement learning (RL). Their system does not rely on pre-programmed models or assumptions; instead represents its control policy as a neural network to make bitrates decisions for future video segments solely through observations of the resulting performance of past decisions. The work in [70] proposed a framework called D-DASH, which combines deep learning and reinforcement learning techniques to optimize the QoE. Their work advocates the use of deep neural networks to learn the excellent video adaptation strategies combined with a carefully designed reinforcement learning mechanism to improve QoE. The authors in [91] proposed a non-cooperative game theory-based bitrate adaptation algorithm with the existence of Nash Equilibrium to optimally allocate the limited bandwidth to multi-users to improve their QoE simultaneously to guarantee fairness. Their work also proposed an efficient distributed iterative algorithm to obtain the Nash Equilibrium of the game (between the server and multiple users) with stability analysis.

The computational complexity of these approaches exponentially increases with an increase in the total number of video segments. Thus, implementation of these models for online bitrate adaptation would be complex and infeasible for resource-constrained mobile DASH clients (running on smartphones).

Parallel Adaptation

The parallel adaptation scheduler requests and download multiple video segments in parallel either from different video sources or leveraging different network paths. Several interesting research efforts have attempted in the recent literature to improve the overall QoE by downloading video segments in parallel.

The works in [92, 93] proposes a multi-path DASH approach by leveraging the multiple network interfaces employed in the client. In contrast to the HTTP range retrieval mechanism, Liu et al. in [94, 95] proposed robust solutions that request and download multiple segments from different servers utilizing the independent HTTP sessions technique. As an evolution of HAS-based streaming for DASH, the authors in [59] proposed Multiple-Source Streaming over HTTP (termed MS-Stream) by leveraging multiple customized servers for delivering high-quality videos. However, these bitrate adaptation strategies require modification of either kernel or network functionalities at both application and transport layers [96]. Recently, the authors in [97] developed a Distributed Queuing theory bitrate adaptation

algorithm for DASH (DQ-DASH) to download the video segments in parallel for multi-source video streaming (i.e., to leverage the availability of video segments at multiple servers). The DQ-DASH bitrate adaptation scheme considers a queuing theory-based bitrate selection to select the best possible representation level and a request scheduler to maintain desired buffer occupancy level subjected to download throughput from geographically distributed servers. Of late, the authors in [98] realized a multi-path video streaming algorithm in the application layer, where each layer of the SVC segment can be fetched using only one of the orthogonal network paths. They have formulated the video segment bitrate decisions as a non-convex discrete optimization subjected to the available bandwidth of the different paths, video segment deadlines, and network link preferences. Accordingly, they proposed a quadratic complexity algorithm for selecting the appropriate segment bitrate, which is shown to be optimal in some special cases.

Push-based Adaptation

The Push-based ABR scheduler requests for a single video segment and the streaming server push multiple additional segments by leveraging the server push mechanism feature of HTTP/2 [51], followed by the primary segment response. Mueller et al. in [99] implemented MPEG-DASH over HTTP/2 (i.e., SPDY) and demonstrated its potential benefits and drawbacks. Furthermore, they have also carried out controlled experimental evaluations to investigate the protocol overhead and link bandwidth utilization with different RTTs as compared to the predecessors.

The works in [23, 24, 25] have developed K-push strategies for low latency live video streaming, in which the server sends κ video segments in a push cycle (i.e., per single client request). In low latency live video streaming, having a smaller segment duration is a straightforward solution to achieve lower live latency. However, this mostly results in an explosion of the HTTP segment requests. In addition, this leads to the inefficient deployment of assets in HTTP caches. The authors in [23] have studied the potential of the server push mechanism to achieve lower live latency and mitigate request rate/overhead in an HTTP/2 based live streaming prototype. The work in [24] proposed a QoE driven adaptive K-push mechanism (QK-Push) for HTTP/2 based live streaming. The QK-Push designed a probabilistic buffer model together with three QoE objective functions for QoE

assurance. The QK-Push formulates the multi-objective optimization problem as a *Pareto optimal problem*. They have used a *Nash bargaining solution* with a *discrete space Lagrangian* method in order to select the appropriate value of κ. They have experimentally evaluated the performance of QK-Push in a controlled network testbed with Planet Lab traces. However, deployment of their approach for deciding the κ value in a push cycle becomes infeasible in a resource-constrained mobile device due to its higher computation complexity. The authors in [100] also investigated the benefits of using HTTP/2 server push for low power video streaming over cellular networks.

The authors in [26] investigated the server push-based mechanism and have demonstrated the benefits of K-push with different values of κ for DASH. Their result analysis uncovers that a static K-push scheme efficiently increases the network bandwidth utilization compared to regular DASH. However, it significantly deteriorates the network adaptability and leads to the video segment "over-push" problem. The over-push problem could further lead to wastage of the network resources such as bandwidth due to viewer abandonment at an early stage. To address the segment over-push problem, the authors in [26, 27] proposed adaptive-push schemes to dynamically adjust group size (κ) for every push cycle based on the network characteristics. The work in [26] proposed an adaptive-push heuristic function called *next-k* to dynamically decides κ value for the next push cycle without degrading the network adaptability. In their approach, a small value of κ is used for the initial push cycle. In all the subsequent push cycles, κ is decided according to the κ value obtained from the previous push cycle while leveraging both the current buffer level and future bandwidth prediction. The authors in [27] designed DASH2M, which aims to dynamically determine adaptation push pair parameters (i.e., (κ, $\hat{b}_\kappa$)) based on the predicted available network resources, the impact of the user's premature termination and the power consumption (i.e., mobile battery consumption). DASH2M first quantitatively decides the number of segments to be pushed in a push cycle to accomplish the goal. Subsequently, it decides the bitrate level for the segments during that push cycle according to the predicted bandwidth. Nevertheless, the predicted bandwidth can deviate from the reality in mobile networks, particularly when the prediction duration is long. The DASH2M also has a push cycle termination phase leveraging the stream termination feature of HTTP/2 to cancel all the upcoming pushed segments during an inaccurate bandwidth prediction. However, the efficacy of the DASH2M

approach is experimentally evaluated by considering only two metrics, namely the perceived video quality variations and battery power consumption on mobile devices.

2.5.2 QoE Estimation Models

QoE modeling has been an active area of research in the academic community for many years. The reliability and accuracy of the QoE model are crucial and of particular interest to the video service providers. In the literature, several QoE models were proposed to predict/estimate the overall user-perceived quality score. In general, these QoE models can be classified using several approaches such as based on the measurement scope; application or service the model applies to; and the stakeholders' scope [101, 102]. In light of the wide adoption, the QoE models are divided into four broad categories, according to the type of input data and the level at which the input information is extracted [4], as shown in Fig. 2.7.

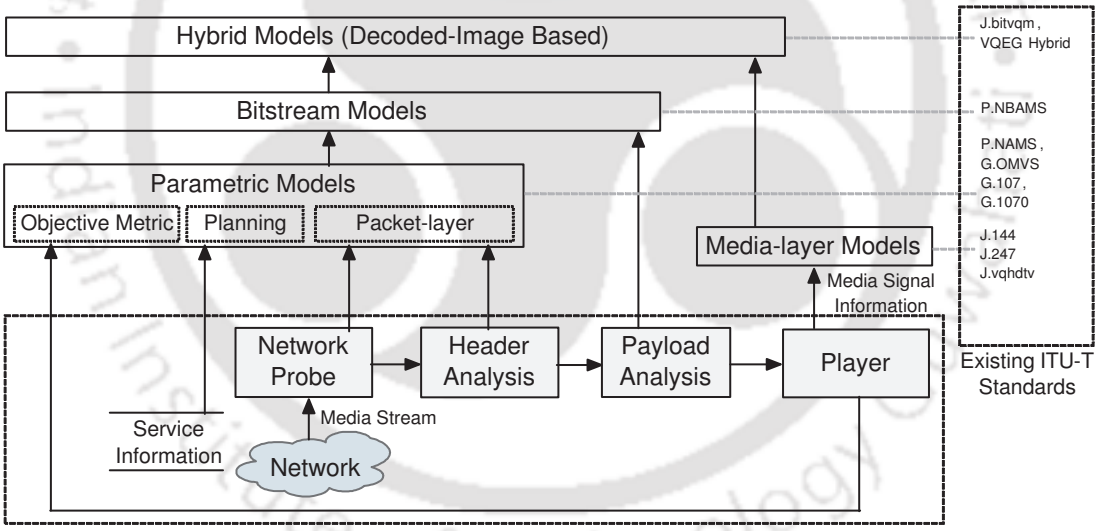


Figure 2.7: Overview of QoE model classification for streaming applications (adapted based on input from [4])

We begin with a high-level overview of the four broad categories of the QoE models. We then describe the QoE models proposed in each of the categories along with their limitations in the rest of the section.

1. *Media-layer Models*: The media-layer models, also popularly known as signal-based models, exploit the actual media signal information to estimate the subjective video

quality. The media-layer models mostly use complex perceptually based psychophysical models, account for low-level visual information processing, to estimate QoE.

2. *Bitstream Models*: The bitstream models take the encoded bitstream and packet layer information such as frame rate, quantization parameter (QP), packet loss, and the motion vector as input to estimate the video QoE.

3. *Parametric Models*: The parametric models estimate the overall video quality by utilizing either service-related parameters or packet-related parameters (i.e., information extracted from packet headers) or application layer objective metrics measured at the client.

4. *Hybrid Models*: Hybrid models are a combination of two or more models mentioned above. Hence, these models take into account more input information as compared to any of the standalone models (as mentioned earlier) to estimate an overall video QoE. Although the hybrid models are more accurate than both parametric and bitstream models, the computation complexity is relatively high and also requires access to the media signals. Thus, limiting their deployment either at the client or at the server-side for QoE monitoring. For brevity, we do not provide a detailed discussion of the hybrid QoE models and refer the interested readers to prior works [4, 22].

Media-layer Models

The media-layer models accomplish the QoE estimation by comparing the degraded output signal to the input signal or just by analyzing the received output signal. Depending on the amount of reference signal information required, these models further can be classified into three broad categories [8]: (i) full reference (FR), (ii) reduced reference (RR), and (iii) no reference (NR). The PSNR [103], SSIM [104], VMAF [105] and ITU-T Recommendations [106, 107], developed originally for image quality assessment, have been some of the widely used FR media-layer models to estimate video QoE. The quality estimation accuracy of these models is relatively high due to the availability of the clean source/reference signal information compared to its counterparts (i.e., RR and NR models). However, in the case of video QoE estimation, these models are ill-suited for larger-scale real-time networks because of their high processing demands. Moreover, the FR and RR signal-based models have to

deal with the non-trivial problem of synchronizing the source input signal to the decoded output signal at the end-point test equipment.

Of late, few media-layer models were specifically developed to predict the QoE for video-on-demand HTTP adaptive streaming system [108, 109]. The work in [109] proposed two QoE prediction models, namely a regression-based evolved PSNR (ePSNR) model and weighted k-nearest neighbors mapping based classification model. The two methodologies consider the multi-segment and multi-rate features of dynamic video sequences with data mining. The regression-based evolved PSNR (ePSNR) model formed by the differential peak signal to noise ratio (dPSNR) statistics and only takes into account the application parameters to predict the QoE. On the other hand, a classification method using the improved weighted k-nearest neighbor mapping based on the video segment positions is proposed to predict QoE. Although the media-layer models have higher prediction accuracy, it has higher computation complexity, especially for longer video sequences with more number of video bitrate levels.

Bitstream Models

The ITU-T Rec. P.1202 (formerly referred to as P.NBAMS) and ITU-T Rec. P.1203 being the most recently approved ITU recommendations for the bitstream-based models. These models have received wider acceptance for adaptive multimedia streaming services over reliable transport [110]. However, the video-quality module in P.1203 is generally not applicable to codecs other than H.264, resolutions above Full-HD. Of late, the ITU-T P.1204.3 [111] provides a bitstream-based model for 4K/Ultra-HD streams together with an open-source implementation.

The authors in [112] proposed a model to estimate the overall quality as a linear combination of median and minimum of the instantaneous quality. They have obtained the instantaneous quality from the quantization parameter (QP) values using an inverted normalized quantization stepsize (NQQ) model [113]. Based on their work, the authors have also observed that the frequency components (of the instantaneous qualities) with the worst quality have the highest impact on the estimated final quality. Tran et al. in [114] proposed a QoE estimation model as a closed-form equation of two influencing factors (IFs), namely encoded video quality and quality variation. They have calculated the encoded

video quality by considering the average quantization parameters and quality variation from histograms of both segment quality values and quality gradients. The authors in [115], as an extension to their previous model [114], also consider three key influencing factors, namely initial startup delay, the perceptual quality value before switching, and interruption related impairments to estimate the overall video quality of a session. Through experimentation, the authors also observed that the impact of the switching amplitude significantly depends on the segment quality value before switching events. Robitza et al. in [116] presented an analytical QoE model, based on the random forest regression of several IFs, for the ITU-T Rec P.1203 competition. Their proposed model averages the per-second scores to predict a final QoE score as experienced by the end-user. They have observed that the length and location of rebuffering events have a significant impact on the prediction accuracy with a clear margin. Despite being relatively computationally inexpensive, the bitstream models are only suitable for the specific codec(s). Thus, making them ill-suited across HAS platforms for QoE prediction.

Parametric Models

The parametric models based on the type of information they utilize to estimate the overall QoE can be further classified into three broad classes and are described as follows.

Planning Models: The planning models estimate the QoE of new networks and services by taking the available service information acquired from the networks and terminals into account during the planning phase. The ITU-T Rec. G.1070 [117] for interactive video-telephony services and ITU-T Rec. G.107 E-model [118] for speech services are some of the most widely used network planning models. Recently, the ITU-T standardized G.1071 [119] (formerly known as G.OMVAS) provides network planning algorithmic models for audio and video streaming applications (including services such as mobile TV and IPTV).

Packet-layer Models: The packet-layer models utilize the information extracted from packet headers only without inspecting the payload information. In both wired and mobile networks, these models capture the quality degradations by detailed monitoring of parameters such as compression, packet loss, and the delay caused due to dropped packets or playback interruptions. Alberti et al. in [120] presented a non-linear parametric QoE model, using a

psychometric model (i.e., polynomial function with rational exponent power), that estimate QoE from the measured values of QoS parameters and their tunable exponents. The authors have reported that the encoding quality mostly impacts the quality degradation on a shorter time interval as compared to IFs, such as quality switching events and interruptions. However, in the absence of the model validation and performance estimation, the actual model performance remains an open question. The authors in [121] proposed a MOS predictor as a linear equation of five functions, which includes the frame rate, total video duration (for which MOS is evaluated), and the total duration of interruptions. Nonetheless, validation of the proposed QoE model using subjective quality assessment is not performed. The authors in [122] devise a model that can map stallings/interruptions to end-user QoE for YouTube based on passive traffic analysis (i.e., relying on packet-level measurements as the only source). The packet-level models are computationally inexpensive and can be deployed easily. However, the absence of any payload information makes them inappropriate for predicting the individual end-user perceived QoE.

Objective Metric-based Models: Unlike other parametric models, the objective metric-based models predict the overall video QoE based on the precise measurements obtained in the client. They can be reported to the model computing location. Parametric objective metric-based models mostly utilize a set of industry-standard application-level objective QoE metrics (as proposed in [21]) such as average video bitrate perceived, number of rebuffering events, and number of bitrate switches. The objective QoE metrics value can be locally measured in the streaming client with an easily deployable lightweight client-side instrumentation script and well suited for real-time application monitoring. Moreover, the quality score estimated by these models can accurately determine the correlation to the individual end-users' viewing experience.

The QoE model presented by Mok et al. [123] was the earliest objective metric-based model to estimate QoE for HAS applications. The work builds the model as a simple linear equation considering three application-level QoE metrics, namely initial loading delay, rebuffering frequency, and duration of rebuffering events. The validation of the proposed model done using only a single video rated by ten users and limited to a single video resolution, which is unrealistic for most HAS applications. The authors in [20] considered five influencing factors (IFs): bitrate switching amplitude, last quality level, frequency

of bitrate switching, time on the highest quality level, and total high-quality playback duration after the last quality switch. They have investigated the effect of the afore-listed five IFs on the QoE. The authors propose a simple QoE model considering only two IFs while discarding the rest of the IFs based on systematic statistical analysis. However, the proposed QoE model (as a parametric equation of two IFs) only considers two video quality levels to account for the effect of switching amplitude and lacks a complete performance validation. Liu et al. proposed a no-reference QoE model [19], adapted from the ITU-T E-model [118]. The model takes a motion vector of both spatial and temporal quality and an estimate based on impairments due to initial delay, rebuffering events, and bitrate switching as input. However, the bitrate switching-related impairments are in turn modeled as the video quality metric (VQM) [124] (i.e., by taking into account both encoding and switching related impairments). The authors carried a preliminary investigation of the model performance using only one-minute long video sequences. Thus, it leaves an open question about the model performance for longer duration video sequences and/or typical real-world HAS applications.

Rodriguez et al. in [18] proposed a video quality model, named Video streaming Quality Metric for DASH ($VsQM_{DASH}$). Their QoE model is built by three stages, (i) each video session is split into three intervals, (ii) modeling the temporal and spatial impairments caused by initial delay, quality variations and interruptions by locally solving the linear equations for each interval, and (iii) obtain the overall quality score by combining the values of the three intervals. In general, the number of switching events during a video session is specifically defined based on segment resolution. However, they have modeled the factor of quality variations in each interval as the frequencies of switching types. Thus, limited information about quality switching is considered for modeling the quality score. The authors in [68] proposed the QoE model in order to achieve higher long-term user engagement, as the weighted sum of the key components: average video quality, average quality variations, rebuffering events, and initial startup delay.

Of late, Yamagashi and Hayashi [17] proposed a parametric quality-estimation model for adaptive bitrate streaming services in TV sets, which was submitted as part of the competition for the ITU-T Rec. 1203. Their parametric quality estimation model only takes audio and video-related application-level information extracted at the client as input and

output per-second respective quality scores, which are then integrated into overall QoE. However, their work has several limitations such as their model is primarily developed for TV sets, lacks validation of the model for smaller screens (i.e., for mobile devices), performance evaluation of only individual quality estimation modules, exclusion of several IFs such as initial startup delay, lack of quality estimation on the high profile codec used for TV sets. Thus, these limitations leave an open question about model performance in real-world scenarios. Wang and Jiang in [85] proposed a QoE evaluation model to evaluate the performance of HAS adaptation schemes based on video segment bitrate level, playback continuity and perceptual fluctuations caused by the bitrate switching events. Of late, the QoE evaluation model proposed in [84] further optimize their earlier model [85] that mainly involve the bitrate switching model. The new bitrate switching model considers the switch amplitude, constant weight coefficients and dividing the video into equal-length short video sequences to eliminate the impact of video length.

2.5.3 Prefetching and Caching Strategies for DASH

In recent years, several bitrate adaptation algorithms have been proposed in the literature leveraging the historical network performance (e.g., throughput), playback buffer occupancy, or a combination thereof to provide an improved QoE (i.e., end-user viewing experience). However, when the network resources (i.e., the backhaul link conditions) or the computing resources (i.e., streaming server processing capacity) are scarce, these bitrate adaptation algorithms could fare poorly. The bitrate adaptation algorithms will either end up with the user experiencing frequent playback interruptions or delivering segments of low-bitrate to end-users so that they have something to watch. From a business perspective, if an end-user has paid a premium price for the service, but it actually fails to provide a satisfactory viewing experience to the end-users, business penalties may be incurred for various stakeholders such as the content provider and/or the network operator in the multimedia streaming process chain. These business penalties, which include a decrease in revenues or a decline in service popularity, could become indispensable differentiators in order to achieve a competitive advantage. To this end, researchers have envisaged more sophisticated strategies to improve QoE for video streaming applications in mobile networks. In this Section, we broadly classify these strategies in the literature into three categories and provide a brief discussion of the

existing works in each of these categories.

Prefetching at Streaming Client

The HAS-based prefetching solutions have been applied to prefetch the video segments to the client devices through recommendations well in advance before the user access. The authors in [125] have proposed a social-aware video prefetching technique supported by the cloud-based agents that prefetch the video segments based on social associations (i.e., online social network interactions) among mobile users. These prefetched video segments are then cached at the client device to provide a better video QoE. However, their proposed approach was mostly focused around H.264 AVC video compression standard of Scalable Video Coded (SVC). Furthermore, it did not address the challenge of prefetching segment(s) with an appropriate bitrate from multiple available bitrate representations. Krishnamoorthi et al. in [126] designed an optimized prefetching policy taking advantage of multiple alternative paths (i.e., making use of parallel TCP connections) to perform the video prefetching at the client. In their approach, the client effectively prefetches multiple segments with a simple round-robin schedule to achieve efficient buffer work ahead in the context of interactive branched video streaming. They have also presented a full implementation of an open-source media framework-based client to provide seamless playback of the interactive branched video. The authors further in [127] present the trade-offs between the importance of ensuring high playback quality for the current video session versus aggressively prefetching new alternative videos. Accordingly, they have proposed an opportunistic prefetching solution to prefetch the segments of a new alternative video(s) when the buffer occupancy of the video being viewed reached beyond a threshold. Thus, their approach with the prefetched segments of new alternative video(s) provides a value-added service of instantaneous playback to the end-users who are more likely to switch to new video(s).

The authors in [128] proposed an adaptation-aware hybrid client-cache prefetching scheme where the client predicts the best possible video bitrate to be prefetched for the next segment. The segment with the client-predicted bitrate is then prefetched leveraging the throughput measurements at the cache. Kim et al. in [129] proposed a streaming component that is based on a quality-aware stochastic optimization approach. Their proposed approach considers a combination of caching on the users' devices and device-to-

device (D2D) communications for reminiscent of HAS video delivery in very dense networks. In their approach, they have considered clients that prefetch a subset of video files from a library and caches them at the client device. Subsequently, the users requesting a video segment (not available in their library) directly obtain it from neighboring users (where it was already prefetched and cached) through D2D communication. The work in [130] proposed a prefetching algorithm to pre-download the videos to the client devices before the user accesses them. Their mobile prefetching algorithm leverages the recommender systems of YouTube to predict the video(s) that are likely to be watched by users in the future. The authors in [131] have proposed a prefetching algorithm leveraging the dynamic programming approach to prefetch data at the mobile terminal. The decision dilemma to (pre)fetch or not in their work is based on the different optimal prefetching policies. These prefetching policies are calculated by taking wireless channel fluctuations, memory constraints, and the application latency into account. Recently, Sengupta et al. in [132] proposed a system, named HotDASH, for opportune prefetching of user-preferred temporal video segments to the client by prioritizing the download of certain chunks over others. The prefetch decision engine, the core of HotDASH, intelligently identifies the prefetch opportunities using a state-of-the-art actor-critic deep reinforcement learning-based model that learns through experience. The trained model of the HotDASH decision engine is then put up as a service, running on a separate server. The DASH client periodically sends the playback information to the service and receives a prefetch decision.

However, the client-based video segment prefetching schemes do not take into account the unique challenges introduced by the end-to-end path that includes both radio access network links and backhaul link conditions on the public Internet. Thus, the streaming client-based prefetching schemes could still experience suboptimal TCP performance. In addition, the effectiveness of the client-based prefetching schemes also relies on the available resources (both computing and storage) at the streaming client, and it could get worse for mobile devices with inadequate storage resources. Several works in the literature focus on either caching or prefetching at the network edge rather than at the streaming client to address the issues mentioned above.

Caching at Network Edge

The research works in this category propose to cache the video segments at the network edge, which are recently accessed by a client. One unique property of the cached content is that the video segment requests are highly concentrated. It is likely that they will be asynchronously and repeatedly requested in the near future. Subsequently, the MEC server will serve the cached video segments directly from the content cache when requests are received from multiple users concentrated around the edge. To improve the cache performance, it has been widely investigated to place the content caching mechanisms at multiple levels within a 4G/5G network architecture. The evolved packet core (EPC) is currently the widely deployed location for content caching mechanisms [133]. Fajardo et al. in [134] proposed an edge caching system leveraging the NFV-based MEC for DASH segment delivery to enhance QoE, especially in mobile multi-user scenarios where clients are accessing the video content asynchronously through a shared radio network. Their work exploits network-assisted adaptive SVC-based DASH where the MEC server (integrated into the eNodeB directly) makes decisions on caching and video adaptation decisions according to the RAN cell statistics and individual channel state information.

The authors in [135] proposed a proactive caching strategy for on-demand videos that caches the segments at the network edge, taking advantage of client's announcements for videos that they will watch in the near future. Ge et al. in [136] proposed QoE-driven MEC caching scheme in order to support MEC-assisted DASH video adaptation. For making the caching decisions, their proposed approach considers a two-level cache replacement strategy that takes advantage of both per-segment, per-quality popularity of videos, and RAN downlink throughput information.

The works in [137, 138] proposed an optimization framework to address the mobile edge caching placement problem for DASH. Their works also proposed a fast approximation algorithm to select the cached representations for multiple videos based on both content information and video popularity under the edge servers' storage capacity constraints. The authors in [139] have proposed a proactive caching strategy for the shared videos in online social networks, which leverages the predicted information of clients' requests and downloads ahead of time at the base stations. In addition, their work formulates the cache placement problem in the next-generation wireless networks (consists of densely deployed low-cost and

low-power small base stations) using a game-theoretic approach called the many-to-many matching game.

The authors in [140], investigate the opportunities to support the video bitrate adaptation algorithm while efficiently utilizing RAN caching, intending to improve both the capacity of the cellular network and video QoE. The proposed joint caching and processing framework selects and cache the best possible highest bitrate video segments from the set of available bitrate levels based on the available cache size, RAN processing resources, and backhaul bandwidth. Subsequently, their approach efficiently performs down-conversion of the video segments to concurrently serve the future client requests requiring a lower bitrate version(s) at RAN caches with limited video processing capability. Nonetheless, their proposed approach may require excessive processing for the transrating (i.e., down-conversion to lower bitrate), with all clients requiring a lower bitrate video and/or over-utilizing the limited RAN processing resource could lead to a poor cache hit ratio over a period of time.

With the objective of introducing cooperative edge caching architecture for 5G networks, the authors in [141] proposed a mobility-aware hierarchical edge caching scheme. In their architecture, the smart vehicles are taken as collaborative caching agents. The smart vehicles' caching resources are utilized for joint scheduling of the edge caching and computing resources to minimizes content access latency. The authors in [142] designed a heuristic ABR-aware proactive cache placement algorithm utilizing the available video content popularity for scheduling of video segment requests at the MEC server in order to minimize the expected content access delay.

However, caching of the video segments at the network edge, especially for HTTP adaptive streaming services, is challenging for multiple reasons. First, the caching scheme performance, i.e., cache hit ratio for HAS services, is determined at the segment/chunk level but no longer evaluated at the video level. With an adaptive bitrate algorithm employed at the client, having a video chunk in the MEC cache does not necessarily translate to having the chunk with the same bitrate/quality level as requested by the client. Thus, designing an efficient cache replacement policy (to cache or evict video segments) becomes more complicated. In addition, it requires dealing with the dual-form of the caching problem, i.e., both at video segment level and bitrate quality level while assuring the QoE to end-users. Second, caching at the network edge works exceptionally well for popular videos, but not all

the segments that belong to the multiple bitrate variants of the videos can be cached at the network edge. Since caching video segments of all bitrate variants could significantly increase backhaul bandwidth and cache storage requirements. In addition, it may also reduce the number of unique videos that can be cached at the edge as the cache may quickly run out of storage and further could lead to degraded cache performance (i.e., poor cache hit ratio). Third, the MEC-based caching approaches for HAS services have little flexibility in adapting to the continuously changing video popularity and client request patterns where segments are requested sequentially and asynchronously. Finally, the unpredictable user mobility in the 4G/5G networks may also heavily affect the MEC-based caching strategies' performance and complicate the content delivery process.

Prefetching at Network Edge

To address the issues with MEC-enabled caching mentioned earlier, researchers have envisaged prefetching strategies at the network edge that will fetch uncached video segments before receiving the actual client request(s). While the prefetching strategies are responsible for moving video contents close to end-users, the MEC servers handle the computation-intensive tasks to seamlessly enhance the end-users' viewing experience. The research efforts for designing segment prefetching strategies have attracted extensive attention in recent times due to its striking advantages in reducing video content acquisition latency, as well as relieving the heavy overhead burden of the network backhaul. In an early work [143], the authors have investigated the performance trade-off between a pair of conflicting performance objectives, namely improving the byte hit ratio and minimizing proxy jitter (i.e., additional access latency incurred for uncached segments) of proxy caching strategies. Based on their insights, they have proposed a prefetching system named *Hyper Proxy* for in-time prefetching of uncached video segments. The authors have also developed a general model to address the conflicting interests between the performance objectives effectively. The model makes two critical decisions: 1) when prefetching is required at a network edge proxy; 2) how many video segments need to be prefetched. However, their approach did not consider the effect of network impairments (especially the round trip time) on TCP performance. Hence, the actual download duration of the segments at the network edge could vary significantly and could lead to degraded performance.

Kleinrouweler et al. in [144, 145] proposed network-assisted DASH architecture based on the software-defined networking paradigm. The authors designed and implemented a network element, called DANE, to monitor concurrent traffic characteristics in the network and DASH media player activity (i.e., segment download patterns). The network element then assists the streaming client by sending recommendations on optimal video bitrates with a goal to improve the viewing experience for the end-users. However, neither caching nor prefetching is involved in their work, which reduces the segment access latency. The authors in [32] proposed to deploy prefetching proxies at the wireless access points (AP) with an aim to improve the overall QoE for the end-users. Their proxy-based content prefetching approach is capable of prefetching the video segment based on the prediction of wireless channel condition at AP. The information obtained from monitoring backbone network traffic and time-varying wireless network conditions is used to decide the next segment's bitrate/quality prior to the prefetch. However, their proposed prefetching approach merely relies on the accuracy of the prediction of the wireless channel conditions. This dependence could lead to performance degradation, including segment misses at the edge in reality, especially under highly fluctuating wireless network conditions. In addition, their proposed approach prefetches only one segment at a time. Hence, it may not be enough to ensure seamless video playback, especially with the backbone networks' unexpected traffic behavior.

The authors in [33], presented a cooperative framework in which clients and proxies share information to perform quality-aware prefetching. However, the prefetching policies in their work also prefetch a fixed number of video segments (i.e., either one segment or a fixed n segments ahead) likely to be requested by the client in the near future. Note that their prefetching policies only prefetch the segments at the same bitrate level as last requested by the client. However, such kind of rigid prefetching policies does not perform well under fluctuating network conditions such as a RAN environment. The work in [146] proposed a collaborative prefetching scheme leveraging the neighboring edge server(s) that aims at preserving session-based user-preferred QoE. However, the prefetching scheme that performs a session-based optimization, rather than a network-based optimization, might able to provide the highest achievable QoE only to a sub-set of end-user sessions where multiple clients share the common last-mile wireless/cellular link. In [34], the authors present a 4K video-on-demand delivery scheme named mobile edge virtualization with adaptive

prefetching with a goal to provide QoE-assured streaming in mobile clients. Their proposed scheme performs adaptive prefetching of the video content on a per-user per-session basis at the edge by maintaining a sufficient gap ahead of the actual client request progress. However, their scheme does not take into account the unique challenges introduced by the DASH applications that include the availability of multiple bitrate variants for each video segment and the video quality switching events made by the bitrate adaptation algorithm employed in the mobile client. Thus, making it ill-suited for QoE-improved DASH video delivery.

2.6 Summary

In this chapter, we first presented a brief background of DASH and QoE, followed by a short description of the MEC architecture. We covered various bitrate adaptation algorithms used in DASH to decide the appropriate video bitrate and enhance the end-user QoE. We provide a broad classification of the exiting the bitrate adaptation algorithms in the literature. Subsequently, we have presented a brief discussion of the works in each of the categories. Next, we explored various QoE estimation models to predict the overall end-users viewing experience for a DASH video session. Based on these QoE models' input to predict the QoE score, we have classified the QoE models into four categories. We have also reviewed the important literature in these categories. Finally, we discussed various important strategies that include caching and prefetching strategies for DASH used to reduce the video segment access latency and improve the end-user QoE in mobile networks.

The discussions till now revealed that several works had pioneered the development of bitrate adaptation algorithms for DASH. While few studies evaluate the performance of the bitrate adaptation algorithms in wired network scenarios, there is no study that explores how the performance of the bitrate adaptation algorithm impacts the end-user overall viewing experience in mobile devices. To begin with, in the next chapter, we assess the performance of client-side bitrate adaptation algorithms using a trace-driven emulation testbed, particularly under time-varying network conditions such as mobile networks. The insights from the quality assessment can help us in identifying the abilities of the bitrate adaptation algorithms in assuring QoE for DASH in mobile devices.

CHAPTER

3

Video Quality of Experience Assessment of DASH Bitrate Adaptation Algorithms in Mobile Devices

This chapter tackles the problem of video QoE assessment of popular client-side bitrate adaptation algorithms in mobile devices. Concretely, this attempt to address the problem: *how bitrate adaptation algorithms could negatively influence the end-user viewing experience and fare in terms of QoE while operating under highly fluctuating bandwidth conditions, particularly in mobile networks?*. To this end, this chapter focuses on video QoE assessment (both subjective and objective assessment) of several popular DASH bitrate adaptation algorithms in mobile devices. The bitrate adaptation algorithms for the video QoE assessment are chosen based on the parameters they consider for segment bitrate selection: (i) rate-based schemes, (ii) buffer-based schemes, and (iii) schemes combining throughput and buffer occupancy. The video QoE assessment is carried out using a quality assessment framework emulating DASH video streaming in mobile devices, under highly variable bandwidth conditions and with several DASH video sequences. The subjective quality assessment results provide clear insights on the viewing experience perceived by the test participants (in terms of rating) for the bitrate adaptation algorithms. The objective video quality evaluation provides a discussion on how the adaptation algorithm's quality decision process determines the actual video bitrate of the segment(s), influences QoE metrics such as playback interruptions, and bitrate adaptation efficiency and video bitrate switches based on the network conditions.

Organization of the Chapter: The remainder of this chapter is structured as follows. We begin with the motivation for performing the video QoE assessment of the bitrate adaptation algorithms in mobile devices in Section 3.1. In Section 3.2, we provide the comprehensive quality assessment framework implementation details, including the DASH server and client implementation, video dataset parameters, and mobile bandwidth traces used for trace-driven emulation. Next, we enumerate the bitrate adaptation algorithms considered for the QoE assessment and provide an overview of their implementation details in Section 3.3. We then present the guidelines for the subjective experiment design and analysis of the results collected from the subjective quality assessments in Section 3.4. In Section 3.5, we define a set of objective QoE metrics and discuss the results obtained from objective video quality evaluation. In Section 3.6, we present the proposed linear parametric QoE scoring model and evaluate the performance of the bitrate adaptation algorithms based on the cumulative QoE score. Finally, we conclude the chapter in Section 3.7.

3.1 Motivation

In recent years, quite a few of the client-side bitrate adaptation algorithms have been pioneered for DASH. Based on the bitrate adaptation algorithm used by the rate adaptation module of the client, the streaming could manifest the following characteristics: smooth vs. agile, conservative vs. aggressive and hence, the user can have different experience [45]. In fact, fixating on a particular algorithm might take it anywhere between the two extremes: low bitrate vs. excessive playback interruptions. Thus, designing a good bitrate adaptation algorithm for adaptive streaming even without worrying about the experience is considered an onerous job [75, 147]. In addition, QoE plays a vital role in the successful deployment of these bitrate adaptation algorithms for online video streaming services. Assuring acceptable QoE is increasingly crucial for both the network providers and multimedia content providers in order to generate revenue. Thus, obtaining a comprehensive list of factors influencing QoE of video streaming services and systematic understanding of their role in meeting the end-users expectations has attracted significant attention in recent times [8, 148].

A handful of measurement studies of popular DASH bitrate adaptation algorithms reported the sub-optimal performance with respect to several QoE measures [14, 15, 45, 75]. The authors of [14] and [15] experimentally evaluated the performance of bitrate adaptation

schemes used for three popular HTTP-based streaming clients, namely Adobe OSMF, Netflix, and Microsoft Smooth Streaming. They have identified noteworthy inefficiencies in bitrate adaptation and reported that none of these bitrate adaptation schemes reacted optimally (i.e., had a relatively longer convergence time) to variations in the network bandwidth. Their analysis also uncovers that some bitrate adaptation schemes just oscillate between the highest and lowest available bitrate levels. Jiang et al. in [45] provide a decent understanding of the problems arising as a consequence of multiple adaptive streaming clients competing for a shared bottleneck link. They demonstrated that all the media players showed a downward spiralling effect under bandwidth fluctuations due to oscillations in bitrate selected for the segments.

Though a substantial amount of work has been undertaken in the design and evaluation of the bitrate adaptation algorithms, to the best of our knowledge, no research has been carried out to understand whether the bitrate adaptation algorithms also ensure good QoE. While QoE is mostly assessed subjectively (through user's explicit feedback), objective metrics that can be measured without user's intervention and are unaffected by user's bias are also gaining importance from a service provider's perspective [149]. Thus, a good bitrate adaptation algorithm not only should select the best possible video bitrate for a given bandwidth but also should ensure fewer interruptions and bitrate switches for ensuring a better viewing experience, considering the fact that these metrics are not independent of each other. These problems could further get aggravated mainly due to the unmanaged nature of bitrate adaptation algorithms in DASH clients that operate in mobile networks with high bandwidth fluctuations, delay variation, and high packet loss rate. In addition, to the best of our knowledge, assessment (either subjective or objective) of the user's QoE with different bitrate adaptation algorithms on mobile devices in the literature has not been carried out.

In this chapter, we attempt to address the question: *how well do popular bitrate adaptation algorithms in the literature ensure end-users QoE as determined either subjectively or collectively by a set of objective metrics, particularly in mobile networks with high bandwidth fluctuations?* In the process, we wanted to see if popular bitrate adaptation algorithms fare poorly in terms of QoE (either subjectively or objectively) while trying to select the best bitrate under time-varying network conditions. To this end, we have performed a

video QoE assessment (both subjective and objective assessments) of several popular DASH bitrate adaptation algorithms in mobile clients. To begin with, we first implement a quality assessment experimental framework with a trace-driven network testbed to emulate realistic mobile client-server path downlink and uplink bandwidth variation scenarios for the video quality assessment. Next, we prepare subjective experimental design guidelines and perform controlled subjective quality experimentation to capture the end-users' viewing experience in mobile devices. We also present an analysis of the controlled subjective quality assessment. We then formally define a set of novel objective QoE metrics to capture the severity of the objective video quality metrics instead of merely measuring them as a numerical value. We perform extensive experiments to collect the quantitative objective assessment results for the bitrate adaptation algorithms with various network trace profile-video combinations, using lightweight client-side instrumentation in the background. Based on the results obtained from the objective QoE assessment, we present a detailed analysis of how the popular bitrate adaptation algorithms fare in the presence of realistic mobile network scenarios, particularly in terms of objective QoE metrics. Finally, in an attempt to represent the overall performance of the bitrate adaptation algorithm as a quantitative measure for QoE (i.e., QoE score), we propose a linear parametric model that considers the observed objective QoE metrics values and their corresponding priorities (obtained from the post-survey measures).

3.2 Quality Assessment Experimental Framework

In this section, we discuss the details of the DASH testbed implementation used for experiments, including the details of the streaming client implementation, video dataset, and mobile network throughput traces.

3.2.1 DASH Testbed Setup

We have deployed an emulated trace-driven network testbed to recreate in-the-wild DASH streaming on mobile devices. The video streaming experiments and QoE assessment were carried out on smartphones running an in-browser DASH media player. The topology of our network testbed is configured as shown in Fig. 3.1. The experimental setup consists of three key components: a web server (for hosting streaming content), a wireless router (configured in 802.11n mode), and a mobile client equipped with a Google Chrome browser. We used an

Apache HTTP Server (version 2.2.24) running on a PC configured with Ubuntu 12.04 OS (kernel version 3.2.0-83-generic). A Samsung i9500 Galaxy S4 smartphone, with an octa-core CPU and 2GB RAM running Android 4.2.2 OS, was used as the streaming client. Several popular DASH adaptation algorithms were implemented in an open-source dash.js player [35] (as discussed in Section 3.2.2). We used a traffic shaping tool called Wondershaper [150], based on Linux `tc` tool, to emulate the client-server path bandwidth variations. The high-speed downlink packet access (HSDPA) network traces were used to emulate the client-server path downlink and uplink bandwidth variation scenarios for the experiments. In Fig. 3.1, the link represented as local indicates a link having effectively unbounded bandwidth and lower link delay. Note that we used a single streaming client in all the experiments.

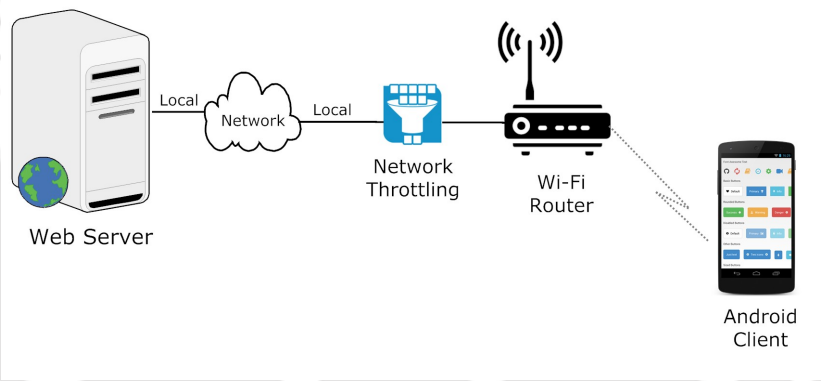


Figure 3.1: Network topology configured in the trace-driven experimental testbed.

3.2.2 Mobile Client Implementation

In this section, we present the implementation details of the bitrate adaptation algorithms in the `dash.js` framework [35], an open-source JavaScript-based browser-independent DASH media player. We used the `dash.js` master branch (v2.1.0 release), the stable version at the time of development. However, this implementation can be easily adapted to future versions of `dash.js` with minimal changes. Note that our implementation does not require any modifications to the traditional web server.

Overview of dash.js: The key modules of the `dash.js` media player are illustrated in Fig. 3.2. The current `dash.js` framework implementation separates video streaming functionality from the specific DASH standard related modules. As we focus on "modify/extend" functions related to adaptive streaming, we left the DASH related modules unmodified. The

key classes and functions responsible for bitrate adaptation in our implementation are listed as follows:

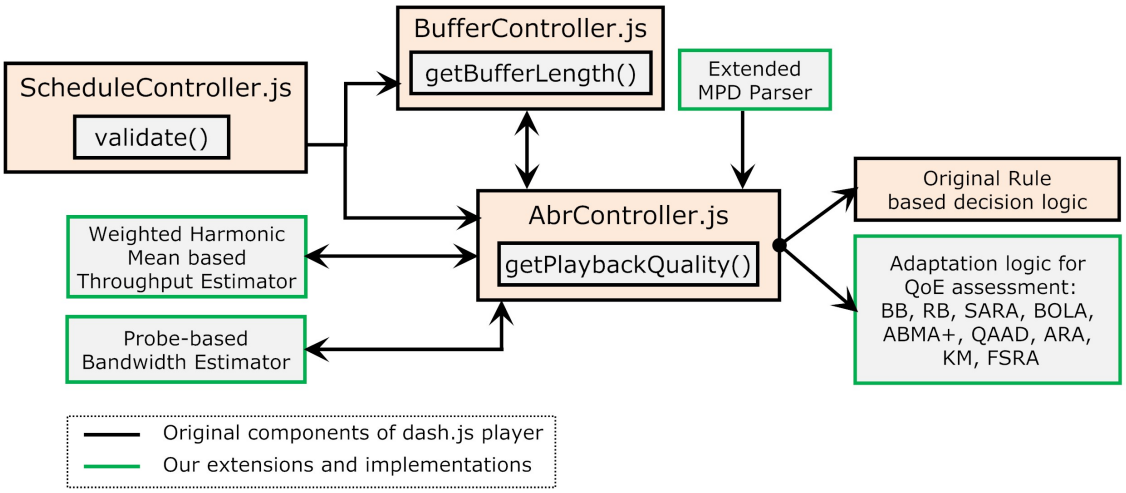


Figure 3.2: A standard dash.js framework code structure and our implementations.

- **ScheduleController:** This class provides a `validate` function, which is periodically invoked to request new video segments. The `validate` function in turn calls the `getPlaybackQuality` function in the `AbrController` class to select a suitable video bitrate for the next segment.

- **BufferController:** The functions provided by this class manage the buffer level and associated events with a goal to avoid playback interruptions. To record the current buffer occupancy, the class also maintains a variable `bufferLevel`. The `bufferLevel` variable is updated at regular intervals by invoking the `getBufferLength` function. The updated `bufferLevel` value is used by the functions in `AbrController` class for making informed bitrate decisions.

- **AbrController:** The functions in this class are primarily responsible for the bitrate adaptation and use a rule-based decision logic to determine an appropriate video bitrate for the segment(s). At the beginning of the video session, the rule gets initialized in the `AbrRulesCollection` class. In the `dash.js` master implementation five rules, namely `ThroughputRule`, `InsufficientBufferRule`, `DownloadRatioRule`, `bufferOccupancyRule`, and `LimitSwitchesRule` have been employed to select the segment bitrate. The `ThroughputRule` selects the video bitrate based on the "average

throughput" (arithmetic mean of the throughput obtained for all the downloaded segments). The `bufferOccupancyRule` make the decisions based on the buffer level and buffer states. It always tries to select a higher bitrate for the segment if enough buffer level is observed. The `DownloadRatioRule` makes use of the "download ratio" (play duration of the last segment divided by the time taken to download the segment) to select bitrate for the next segment. The `InsufficientBufferRule` selects the video bitrate to make sure that the media player buffer would not run dry; in order to avoid the playback stalls. On the other hand, `LimitSwitchesRule` maintains an `qualitySwitchThreshold` variable, initialized at the beginning of the video session, to limit the number of bitrate switches while allowing the stream to settle down. In the `AbrRulesCollection` class priorities are assigned to each of these rules to resolve the conflicts and make final bitrate decision(s).

Challenges and Modifications in dash.js: We observed two challenges in the current `dash.js` implementation for integrating the DASH adaptation algorithms. First, the `validate` function is invoked periodically to update the buffer level and, in turn, invoke the functions for the bitrate adaptation in the `AbrController` class. Periodically invoking the functions in the `AbrController` class clearly indicates that the bitrate decision(s) in the current implementation is not always made once for each segment. Thus, periodically invoking the `AbrController` class functions could even lead to fetching the same segment with different bitrate levels. Next, multiple segments are being downloaded without prioritizing the segments requested earlier in the stream. This may lead to an inaccurate estimation of the download rate for a segment. To address these challenges, we made two changes to the `ScheduleController` class in the `dash.js` code as discussed further.

1. We invoke the `AbrController` class functions to choose the bitrate(s) only at the start of each segment. This addresses the problem of downloading the same segment with multiple bitrate levels.

2. We made custom changes to download the video segments sequentially. We decide the bitrate for the next segment only after the current segment gets completely downloaded.

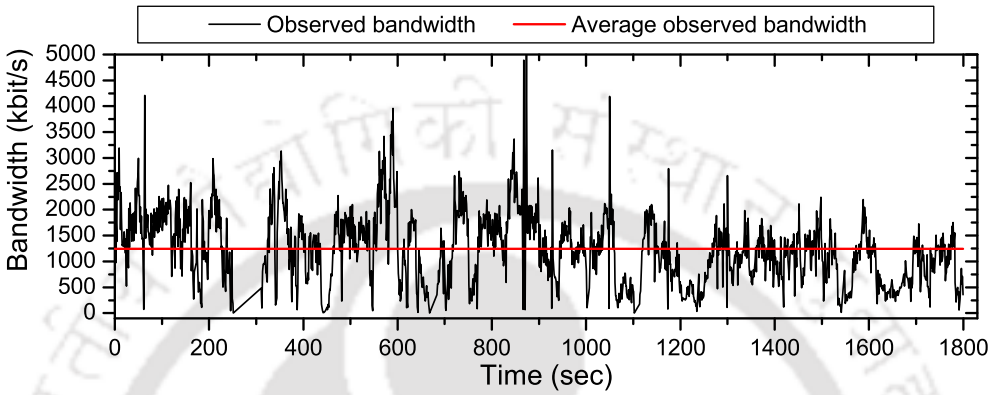
These modifications to the `dash.js` code are also consistent with all the state-of-the-

art bitrate adaptation algorithms considered for the video QoE assessment (discussed in Section 3.3). To implement different bitrate adaptation algorithms (e.g., SARA, buffer-based, BOLA, etc.), we extended the basic implementation by replacing the existing rule-based bitrate adaptation logic with custom adaptation logic (as shown in Fig. 3.2). The segment size information plays a vital role in the bitrate selection by the SARA algorithm (as discussed in Chapter 2.5.1). To this end, we have extended the MPD parser module in `dash.js` to read the segment sizes from the modified MPD file. In SARA implementation, we have implemented a weighted harmonic mean based throughput estimation scheme. For FSRA implementation, we have implemented a probe-based bandwidth estimation scheme. We have modified the `AbrController` class to the log playback buffer evaluation, number of interruptions, duration of the interruptions, number of bitrate switching events, bitrate selected for the segments, and the estimated throughput during the video session.

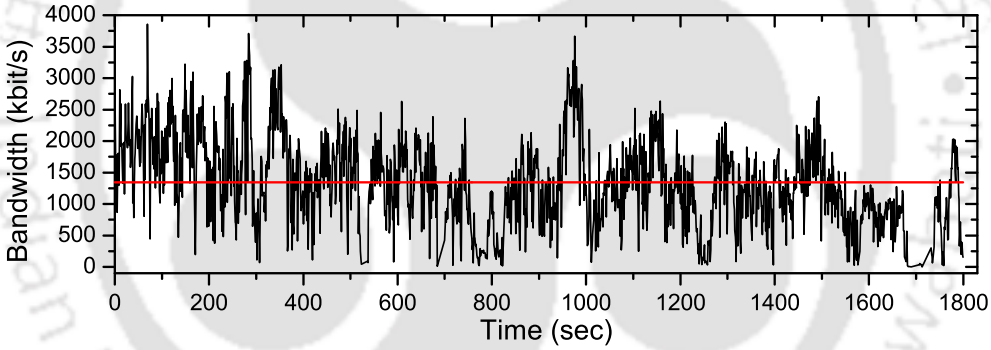
3.2.3 Mobile Throughput Traces

The performance of QoE assessment is closely correlated with the underlying network bandwidth conditions. Unlike wired networks, mobile networks are mostly characterized by their intense throughput variations. Additionally, the prolonged low bandwidth due to diverse coverage quality could lead to an increased probability of playback interruptions and frequent bitrate switches. To perform the quality assessment under challenging bandwidth fluctuations as observed in real-world networks with mobility, we have used the throughput traces from the HSDPA dataset [151, 152]. The throughput traces in the dataset were measured in Telenor's 3G/HSDPA mobile network across popular commute routes in Norway. The trace consisted of 30 minutes of contiguous 1 second application-layer throughput measurements recorded with a video streaming client. We considered three throughput profiles for our experiments, which correspond to direct throughput measurements collected while commuting on bus, car, and ferry from the full dataset. The throughput profile of traces collected across train, car and ferry are shown in Fig. 3.3a, Fig. 3.3b, and Fig. 3.3c, respectively. These traces had sufficient variation in the average throughput and varied in terms of their throughput profile. The significant distinction of throughput profile across the three traces is mostly due to mobility characteristics (i.e., the speed at which the device is traveling) and their measured location. These traces were useful for stress testing of

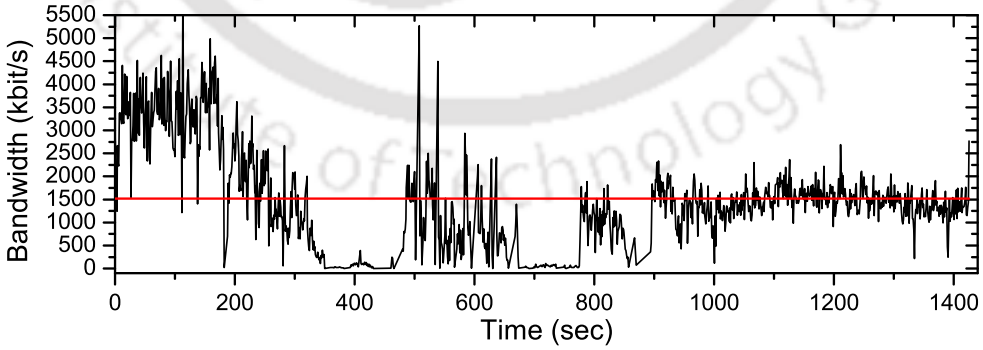
the adaptation algorithms since the throughput was measured on short intervals (1 second intervals).



(a) Train Trace Profile



(b) Car Trace Profile



(c) Ferry Trace Profile

Figure 3.3: Pictorial illustration of throughput trace profiles collected while commuting on: (a) Train (b) Car and (c) Ferry.

3.2.4 DASH Video Dataset

To perform the quality assessment, we have selected three representative open movie sequences from the DASH dataset [153], as recommended in the measurement guidelines of the DASH Industry Forum [154]. The first movie is Big Buck Bunny (BBB), a high-motion computer-animated movie. The second is Red Bull Playstreets (RBPS), a sports documentary with high-motion scenes. The third is Tears of Steel (ToS), which is an SCI-FI genre movie with regular motion scenes. The video resolution and encoding rate for the three videos' bitrate levels are summarized in Table 3.1. We considered eight representation/bitrate levels for all three test video sequences. The number of bitrate levels is consistent with the levels used by YouTube [155]. Additionally, considering a moderate number of distinct bitrate levels would lead to a smaller probability of frequent bitrate switching events during the playback. We have restricted the total video length to about 480 seconds, with 48 segments of a play duration of 10 second. The segment play duration (i.e., 10 seconds) is consistent with the range of chunk durations that are widely used in real-world commercial services [156]. Although segments with different play durations could be considered, we found that this factor does not significantly affect the video quality assessment.

Table 3.1: Summary of DASH video resolution and encoding rate (in Mbps) used for video quality assessment.

RI	**BBB**		**RBPS**		**ToS**	
	MER	Resolution	MER	Resolution	MER	Resolution
1	0.045	320x240	0.10	320x240	0.25	480x270
2	0.177	480x360	0.25	480x360	0.50	640x360
3	0.317	480x360	0.39	480x360	0.81	640x360
4	0.569	854x480	0.89	854x480	1.50	1280x720
5	0.987	1280x720	1.50	1280x720	2.40	1280x720
6	1.40	1280x720	2.50	1280x720	3.00	1920x1080
7	2.40	1920x1080	4.00	1920x1080	4.00	1920x1080
8	3.50	1920x1080	5.90	1920x1080	6.00	1920x1080

MER: Maximum encoding rate (in Mbps)
RI: Representation Index

3.3 Bitrate Adaptation Algorithms Considered

For video quality assessment purpose, we consider nine popular DASH bitrate adaptation algorithms belong to the three different classes, namely rate-based, buffer-based and a combination thereof, to study their impact on end-users QoE in mobile devices. We presented a brief discussion of these bitrate adaptation algorithms in Chapter 2.5.1. The stochastic-based adaptation schemes (such as MDP-based [67], MPC-based [68] and RL-based [36] schemes) were intensive in computation and storage requirements (e.g., as MPC uses require pre-computed tables). Hence, it was difficult to implement these adaptation schemes in resource-constrained mobile phones. For real-time bitrate decision making, we found the algorithms from the three classes as mentioned earlier useful and hence, we consider them only for the video quality assessment.

In the rest of this section, we enumerate the nine bitrate adaptation algorithms together with an overview of their implementation details for completeness and clarity. We make the bitrate adaptation algorithms implementation code[1] publicly available for reproducibility. In addition, we consider only three bitrate adaptation algorithms (as marked in Table 3.2) only for subjective quality assessment as including more algorithms causes fatigue to the subjects during the subjective quality assessment.

Table 3.2: Summary of DASH adaptation algorithms considered for objective and subjective quality assessment together with the set of features they employ for bitrate adaptation.

Adaptation Algorithms	**Objective Quality Assessment**	**Subjective Quality Assessment**	**Features Employed for Bitrate Adaptation**
Buffer-based (BB) [76]	✓	✓	Playback buffer occupancy (in seconds)
Rate-based (RB) [71]	✓	✓	Historical network throughput
SARA [39]	✓	✓	Current playback buffer occupancy, estimated network throughput and impact of segment size variation on the measured throughput
BOLA [37]	✓	✗	Current buffer level (measured as the number of segments)
ABMA+ [40]	✓	✗	Predicted video playout rebuffering probability
QAAD [87]	✓	✗	Current buffer occupancy and the estimated network bandwidth
ARA [80]	✓	✗	Estimated network throughput and current buffer occupancy (in seconds) along with four scaling factors to provide fastest response time
KM [79]	✓	✗	Estimated network throughput along with current buffer occupancy (in playback seconds)
FSRA [74]	✓	✗	probed bandwidth and buffer occupancy

[1] https://github.com/hemakyarnagula/DASHAlgorithmCode/

1) **Buffer-based algorithm (BB):** The buffer-based algorithm merely considers the client's current playback buffer occupancy (B_t, in seconds) as a marker for adaptation. We employed the rate map function ($f(B_t)$) suggested by Huang et al. in [77] to map the buffer occupancy with the set of available representations. This algorithm selects a maximum available bitrate that is less than or equal to the discrete bitrate suggested by the rate map for fetching the next segment. In our implementation, we have used the rate map with a buffer reservoir of 5 seconds and a maximum buffer capacity of 60 seconds.

2) **Rate-based algorithm (RB):** We employed the function as suggested in [71] for the throughput prediction in RB. This algorithm considers the historical network throughput as a marker for segment bitrate adaptation. The maximum available bitrate close to the predicted throughput is selected as an appropriate bitrate for the next segment.

3) **SARA:** The segment-aware rate adaptation (SARA) algorithm presented in [39] was implemented and the MPD files were enhanced by adding the segment size information. In addition, we extend the MPD parser module to read the segment sizes from the modified MPD file. The SARA algorithm considers the impact of the segment size variation on the measured throughput and current playback buffer length along with the estimated network throughput to decide an appropriate bitrate for the next segment (as discussed in Chapter 2.5.1). For our implementation, the SARA buffer thresholds were configured as $I = 2$, $B_\alpha = 5$, $B_\beta = 10$, and $B_{max} = 12$. These buffer thresholds used for buffer occupancy decide the size of the playback buffer reservoir (in seconds).

4) **BOLA:** We employed the function suggested in [37] for the Buffer Occupancy based Lyapunov Algorithm (BOLA). The BOLA for bitrate adaptation makes a control decision by merely utilizing the current buffer level (measured as the number of segments). Besides, the BOLA algorithm does not require explicit prediction of the available network bandwidth for making the control decisions. In our implementation, we set the input weight parameter values for prioritizing playback utility (γ) and control parameter related to the maximum buffer size (V) to 0.5/segment play duration

and 0.93, respectively.

5) **ABMA+:** The adaptation and buffer management algorithm [40], called ABMA+, merely considers the predicted video playout rebuffering probability as a marker for adaptation. The ABMA+ algorithm uses a precomputed buffer map to avoid heavy on-line computations. We employed the precomputed buffer map suggested by Beben et al. [40] to determine the size of the playback buffer, which in turn ensures a given rebuffering probability. In our implementation, we have assumed the rebuffering probability threshold equal to 10^{-4}, anti-oscillation factor equal to 0.1, and a buffer capacity of 60 seconds.

6) **QAAD:** The QoE-enhanced adaptation algorithm over DASH (QAAD) [87] employees a fixed estimation interval period which is larger than RTT in general and smaller than the segment play duration for bandwidth estimation. Subsequently, it considers both the current buffer occupancy and the estimated network bandwidth to avoid interruption and minimize video bitrate switching amplitude during the video playback. For our implementation, we set the estimation period θ and the weight factor for sampled bandwidth ω (to smoothen the estimated bandwidth by means of a weighted moving average scheme) as 0.3 seconds and 0.875, respectively. We also set the predefined marginal buffer length and the minimum buffer length values to 10 seconds and 3 seconds, respectively.

7) **ARA:** The advanced rate adaptation (ARA) algorithm [80] leverages four scaling factors with an objective to provide the fast reaction time at stable network conditions and stabile adaptation regardless of underlaying network condition (see discussion in Chapter 2.5.1). For our implementation, we initialize the values for the four buffer levels, namely B_{Min}, B_{Low}, B_{High}, and B_{Opt} as $2 \times SegmentDuration$, $6 \times SegmentDuration$, $10 \times SegmentDuration$, and $(B_{Low} + B_{High})/2$, respectively. We also consider the values 0.3, 0.5, 0.7 and 0.9 (chosen experimentally in [80]) for the four scaling factors: $\alpha 1$, $\alpha 2$, $\alpha 3$ and $\alpha 4$, respectively in order to provide the fastest response time without jeopardizing the stability.

8) **KM:** As proposed by Miller et al., the KM algorithm in [79] uses an estimated network throughput along with current buffer occupancy (in playback seconds) to decide a

suitable bitrate for the next segment. The KM algorithm maintains three buffer thresholds, namely B_{min}, B_{low}, and B_{high} (in seconds of playback time) to improve the robustness of the algorithm due to network measurement uncertainties. In our implementation, we set the values of the three buffer thresholds B_{min}, B_{low}, and B_{high} to 10, 20, and 60 seconds, respectively.

9) **FSRA:** The fairness-aware smooth rate adaptation (FSRA) approach considers a dual-threshold based adaptation scheme combining the playback buffer occupancy and the probed bandwidth. We employed the probe-based bandwidth estimation scheme as suggested in [74]. The probe-based bandwidth estimation includes two schemes: a logarithmic law-based increase probing scheme (to avoid over-probing) and a conservative back-off based decrease probing scheme (to avoid congestion). In addition, it also uses two operation thresholds to mitigate the effect of bandwidth variations and obtain the best video bitrate for the future segment(s).

3.4 Subjective Video Quality Evaluation

In this section, we present the results from the subjective QoE assessment. We first describe the subjective metrics used in the experiments to assess the QoE with the users watching the video on a mobile phone. In addition, we discuss the assessment method and experimental design to conduct the controlled subjective experiments.

3.4.1 Subjective QoE Metrics

In general, the subjective experiment's purpose is to collect multiple subjects ("participants") opinions about the service performance. We have used the Absolute Category Rating (ACR) method to record the subject's true perception of watching the DASH video during the assessment. The ACR is a single-stimulus continuous quality evaluation method outlined in the ITU-T recommendations [49]. The ACR is the most widely accepted method to evaluate the quality of video services. In this method, the test participants are asked to passively watch the test video sequence(s) that is/are presented one at a time. The participants are then asked to provide their overall viewing experience individually without comparing the test video sequence(s) to an explicit reference. We have used a rating scale

with labels based on the five-point scale, ranging from 1 to 5 with labels; "5 - Excellent", "4 - Good", "3 - Fair", "2 - Poor", "1 - Bad". In contrast to other rating scales (e.g., a nine-point scale or an eleven-point scale), the choice of a five-point rating scale is mainly to suit the client device's smaller display size. Fig. 3.4 illustrates an experience rating interface used for collecting the participants' overall viewing experience in the mobile phones. The mean opinion score (MOS) is the most popular and the de facto metric for subjective QoE assessment. The MOS is a rational number that simply averages all the individual subject opinions/ratings reported from the experiments. We have used MOS to assess the subjective QoE for all the video presentations considered. For each of the video presentations, we have calculated 95% confidence intervals (CIs).

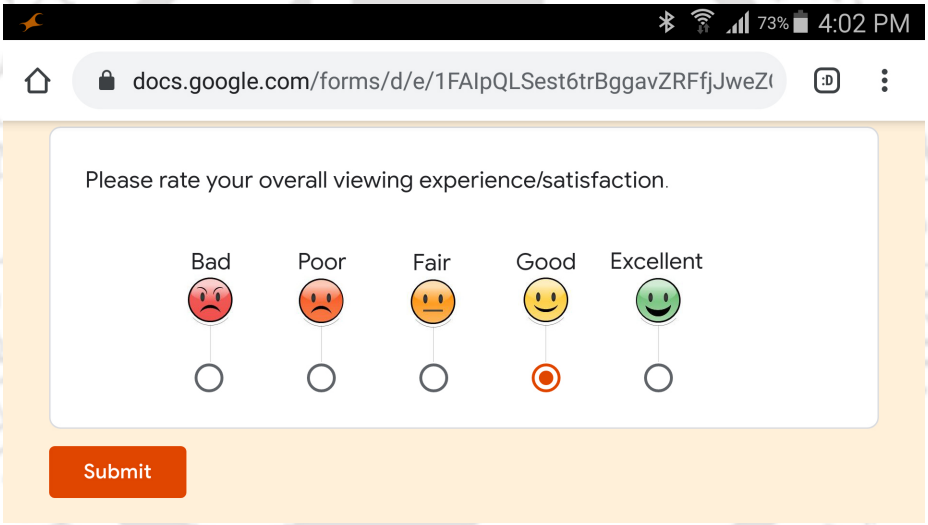


Figure 3.4: Screenshot of an experience rating scale used for the collecting viewing experience.

3.4.2 Subjective Assessment Method and Experimental Design

We conducted controlled experiments with test participants subjected to a controlled viewing experience to assess the end user's perception and reaction. We have used three popular DASH adaptation algorithms, namely rate-based [71], buffer-based [76], and SARA [39] for subjective assessment in mobile devices (as discussed in Section 3.3). We have used the short animation movie sequence *Big Buck Bunny* and three bandwidth trace profiles from the network trace dataset for experimentation (as discussed in Section 3.2.3). Thus, we have nine test scenarios to be evaluated, based on a 3 × 3 factorial design experiment that

reflects combinations of network trace profile and the DASH bitrate adaptation algorithm.

Participants: We invited eighty-nine non-expert test participants/subjects to watch the video content on smartphones with either one of the three trace profiles. We then asked the test participants to convey their experience as a subjective score. The subjective quality assessment was carried out with participants at the Indian Institute of Technology Guwahati (IIT Guwahati), India and the University of Missouri-Kansas City, Kansas City, United States. The test participants, predominately, are either university graduates or undergraduate students. The participants volunteered to take part in our experiments without any incentive. The subjects had no knowledge about our algorithm implementation or their usage with the video sequence. The participants were not directly concerned with audiovisual quality as a part of their work; therefore, they were not experienced assessors. In terms of representativeness, we have summarized the test participants' demographic features collected through an exit survey in Table 3.3. Despite the attempts to obtain a diverse test participant group, we observe from Table 3.3 that the participants are still skewed toward younger age.

Table 3.3: Summary of test participants demographic features collected during exit-survey.

Demographic	**Participants**
Young participants (% 19-32 years old)	55.68
Gender (% male)	68.96
Smartphone usage (in years)[a,b]	4.862 (1.331)

[a] An average of 'number of years' the participant has used a smartphone for watching videos.

[b] Means also include standard deviations.

Experiment Procedure: Prior to the experiment, in terms of procedure, each test participant was provided with instructions (text only) to ensure they had all the necessary information. The participants were given sufficient time to understand the procedure followed to rate their video session experience. The participants were asked to clarify their doubts before starting the experiment to avoid experimenter opinion bias. Attending all nine test scenarios could be strenuous for the participants and difficult to retain interest for the entire duration of the experiment. In addition, this could also produce unwanted biases due to any carryover effects from the previous session. To address this issue, we ensure that each test participant attends only one session with video streamed using one

of the bandwidth trace profiles chosen randomly. For each participant, the session would last approximately 30 to 40 minutes. The session consisted of three video test scenarios (each of 8 minutes play duration). Each of these video test scenarios uses one of the three adaptation algorithms used for subjective assessment. No test scenarios use the same bitrate adaptation algorithm twice within a session. The participants were not allowed to pause the video. However, we give a short break of a few minutes after watching the video in a test scenario to reduce the subjects' fatigue. The ACR stimulus time pattern used in the experiments is illustrated in Fig. 3.5.

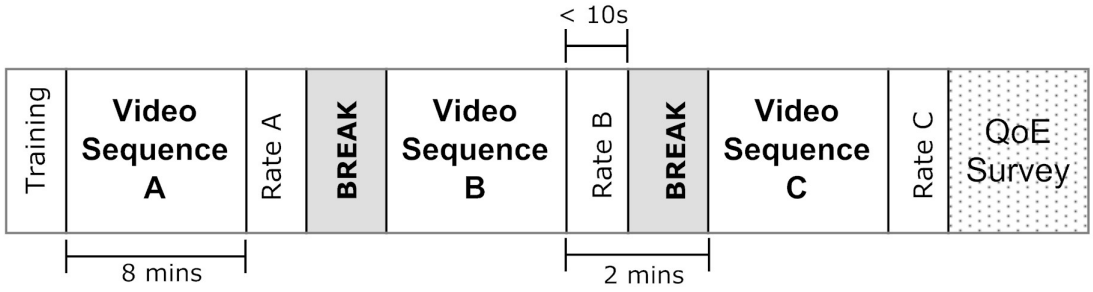


Figure 3.5: Illustration of ACR stimulus time pattern used in subjective assessment.

The controlled ambient lighting conditions were maintained according to the ITU-T recommendations [49]. Besides, we allowed the participants to use the client device normally and freely adjust the viewing distance to get a full view of the screen. Thus, facilitating the most natural/realistic viewing conditions. All the test participants had normal visual acuity and color vision (or wore corrective glasses). To avoid order bias in the experiments the video clips were played in the same sequence for all the subjects [157]. The subjects were simply asked to convey their overall viewing experience as a score on the five-point ACR scale (discussed in Section 3.4.1). We instrumented the client with a rating interface to record the subjective score/rating immediately after watching the video in a test scenario (as shown in Fig. 3.4). At the end of each video viewing session, the participants were asked to fill out a post-survey (or exit survey). The post-survey goal is to obtain quantitative information that collectively indicates the participants' perceived experience and their preferences/expectations (refer Section 3.6.1).

3.4.3 Qualitative Subjective Assessment Results and Discussion

We performed controlled subjective quality experimentation to capture the end-users' viewing experience in mobile devices. The quality of the subject experience ratings is potentially influenced by unreliable participants and could overly influence the results. In general, an outlier is defined as an observation that is generated from a model or captured during subjective experimentation and was different from the main "body" of data. The earlier mentioned definition implies that an outlier may be found anywhere within the range of observations. However, it is natural to suspect and examine only the extreme observations/values in the dataset as possible outliers. This practice is quite popular and commonly used in the field of Analytical Chemistry for data treatment, wherein the datasets sometimes comprise one suspect value that is significantly higher or significantly lower than the other values. Moreover, including an outlier value in the data affects calculations like the mean and standard deviation. Thus, true outliers in the dataset should be removed. The rejection of suspected observations in the dataset must be based exclusively on an objective criterion and not on subjective or intuitive grounds. This can be achieved by using statistically sound tests for "the detection of outliers".

Accordingly, we legitimately remove potentially unreliable experience ratings (i.e., outliers) by filtering out the ratings using appropriate outlier detection techniques. Based on the recommendations in [158], we first screened the subjective ratings/scores collected for each of the trace-algorithm combinations for outliers. We choose Dixon's Q Test [159] to detect and reject outliers from the collected subjective rating/scores.

The Dixon's Q Test, often simply referred to as the "Q test", is a simpler statistical test that is commonly used for identification and rejection of outliers from a very small dataset conforming to a normal (Gaussian) distribution. Q test is based on the statistical distribution of "subrange ratios" of ordered observations. Hence, a normal (Gaussian) distribution of data is assumed whenever Q test is applied. The Dixon's Q Test is equally well suited to larger datasets if only one outlier is present. This test allows us to examine if one (and only one) observation from a small set of observations (typically 3 to 30) can be "legitimately" rejected or not. It can be used to test whether the maximum value is an outlier, the minimum value is an outlier, or either the minimum or maximum value is an outlier. In case an outlier is detected and rejected, Q-test cannot be reapplied on the set of

the remaining observations. To apply Q test on our collected subjective rating/scores, we follow the procedure as itemized below:

- The N values comprising the dataset under examination are arranged in ascending order (i.e., from the lowest to the highest value).
- Subsequently, the statistic experimental Q-value (Q_{exp}) is calculated. The Q_{exp} is defined as the difference of the suspect value from its nearest one divided by the range of the values (Q: rejection quotient).
- The obtained Q_{exp} value is compared to a critical Q-value (Q_{crit}) that corresponds to the confidence level (CL) decided to run the test (usually: CL = 95%).
- In case of obtained Q_{exp} value is greater than the critical Q-value (i.e., $Q_{exp} > Q_{crit}$), the suspected value can be characterized as an outlier and can be rejected. If $Q_{exp} < Q_{crit}$, the suspected value must be retained and used in all subsequent calculations.

To detect and reject the outliers from the subjective ratings/scores collected for each of the trace-algorithm combinations, we have considered a critical Q-value (Q_{crit}) of 0.361 with 30 subjective ratings/scores and at 95% confidence level (i.e., at an alpha level of 0.05) for the two-tailed Q test from the tabulated data in [160, 161].

The screening test reported one video session each, from two participants as an outlier. We have also verified whether the subjective ratings differ by more than twice the standard deviation from their corresponding mean values for these two outlier cases. Since this was found to be the case, we excluded all the six video sessions rated by the two participants from the results. Thus, we have a total of eighty-seven test participants who watched the video content on smartphones and rated their viewing experience as a subjective score. Apart from this, we also ensure a sufficient and equal number of participants attended for each of the video-trace combinations (as recommended by ITU-T [158]). Thus, one-third of the test participants (i.e., 29) watched the video content with train, car, and ferry trace, respectively. A total of 261 video sessions (from eighty-seven participants) of various network trace profile and adaptation algorithm combinations have resulted from the subjective quality experimentation.

We have observed disparity in the test participants' individual rating, mostly due to the differences in participants' judgment. We have computed the mean opinion scores (MOS) and associated 95% confidence intervals (CIs) from the individual subject ratings for each of the trace-algorithm combinations to address this. From our results, we noticed that visual discomfort during streaming severely affected the score given by the participants. The main reason for visual discomfort is attributed to either a lower average bitrate played or frequent interruptions. The CDF for the MOS scores obtained for all the nine trace-algorithm combinations is plotted in Fig. 3.6. We have also summarized MOS for nine trace-algorithm combinations in Table 3.4. The Fig. 3.6a, Fig. 3.6b and Fig. 3.6c show the CDF of MOS with the bandwidth emulated from the train, car and ferry traces, respectively. From Fig. 3.6, we observe that higher MOS recorded by the SARA algorithm guarantees the best QoE among all the adaptation algorithms. The rate-based adaptation was always given a poor rating by the subjects, mainly due to playing segments with lower bitrate levels and frequent interruptions. We have also observed that both the SARA and the BB algorithms fetched video segments with higher bitrates by adapting to the current network conditions. However, we have also observed that the BB algorithm fetched segments of higher representations aggressively merely based on the buffer occupancy (without considering the current undelaying network conditions), leading to interruptions of longer durations (reported later in Table 3.6). The longer duration interruptions for the BB algorithm were found to be the main reason for both visual discomforts experienced by the participants and comparatively the lower MOS recorded across all experiments. On the contrary, streaming with the SARA algorithm was smoother because it considers the segment size information before selecting suitable video representation, given the estimated bandwidth and playback buffer occupancy. In all the experiments, participants have given the SARA algorithm higher ratings as compared to the others, mostly due to the overall viewing experience. This has resulted in higher MOS for the SARA algorithm in all the trace-algorithm scenarios (as reported in Table 3.4).

We have plotted the distribution of absolute participant experience ratings together with a mean of 3.107 in Fig. 3.7a. From Fig. 3.7a, we can observe that experience rating distributions are not skewed toward any particular ratings in the absolute case. Thus, the distributions indicate a good spread of participant ratings across the five-point rating scale.

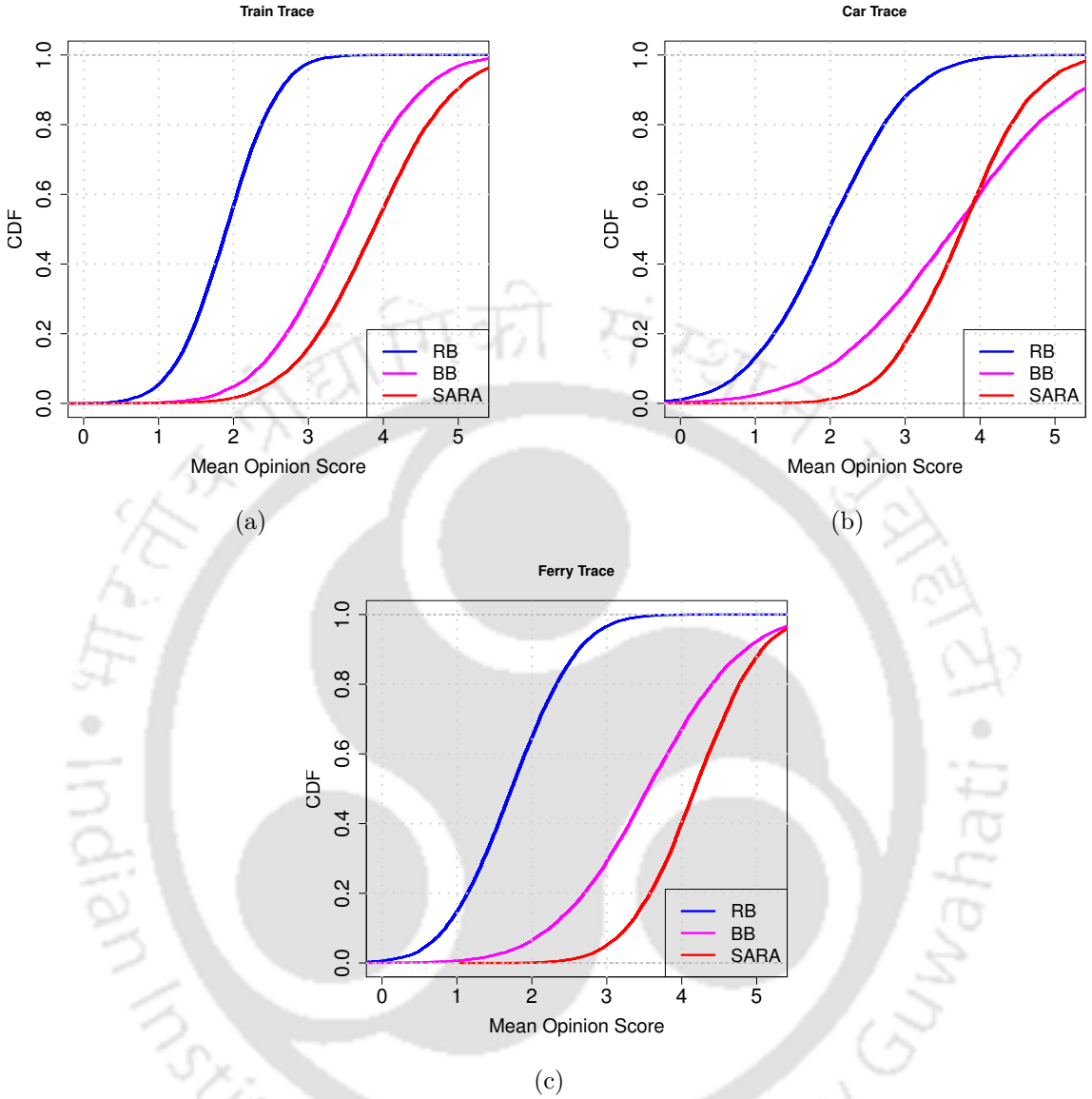
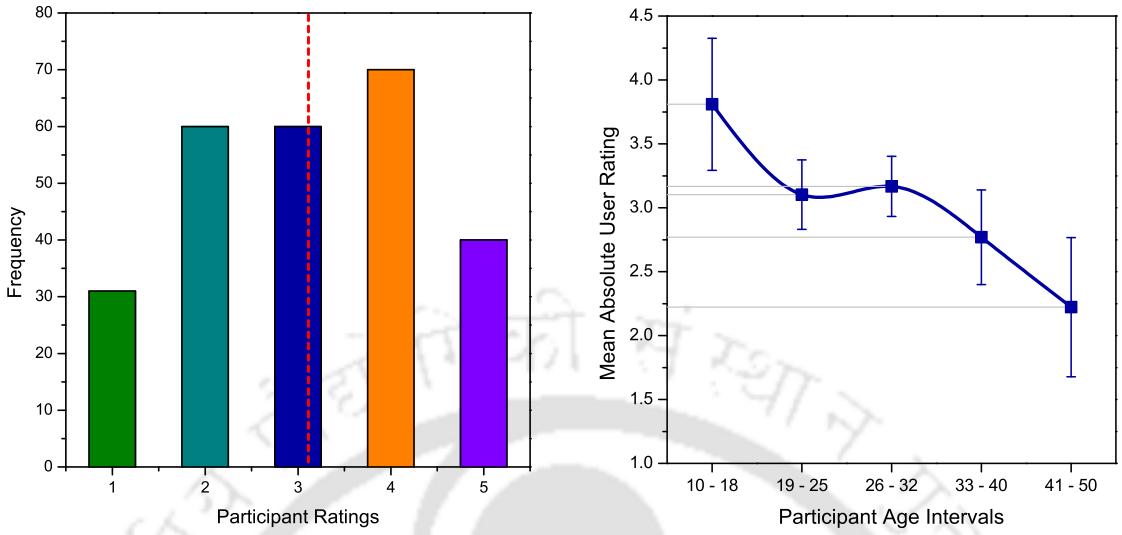


Figure 3.6: CDF of mean opinion scores with different throughput traces: (a) Train (b) Car and (c) Ferry.

At the individual level, we analyze the relationship between the absolute experience rating and participant demographic data (collected from the exit survey). Fig. 3.7b shows the mean absolute rating for different age intervals. It is evident from Fig. 3.7b that the absolute experience ratings tend to decrease with age. This suggests that older participants might be using the five-point rating scale differently (potentially using the "Fair" rating as a baseline instead of the "Excellent" rating). However, we did not observe any significant bias in the absolute ratings based on gender or experience in smartphone usage.



(a) (b)

Figure 3.7: Statistics from subjective assessment: (a) distribution of absolute participant experience ratings (with dashed line representing mean of participant ratings), (b) mean absolute participant experience rating observed for different age intervals.

Table 3.4: The mean opinion score obtained from subjective QoE assessment with Big Buck Bunny video sequence.

Bandwidth Trace Profile	Adaptation Algorithm		
	Rate-based	Buffer-based	SARA
Train Trace	1.897 (0.2184)	3.414 (0.3397)	3.862 (0.3431)
Car Trace	1.966 (0.3326)	3.621 (0.5179)	3.759 (0.3022)
Ferry Trace	1.724 (0.2751)	3.552 (0.4001)	4.172 (0.2785)

Note: confidence interval (CI) for MOS in brackets.

3.5 Objective Video Quality Evaluation

In this section, we present and formally define a set of objective QoE metrics to assess the QoE of the bitrate adaptation algorithms. We then provide a detailed discussion of the quantitative objective assessment results.

3.5.1 Objective QoE Metrics

Objective QoE assessments' essential requirements are gathering, aggregation, and correlation of the measurements to assess overall QoE. The work in [21] proposed a set of industry-standard application-layer objective QoE metrics as an alternative to the network-related

factors. The application-layer objective QoE metrics include but not limited to average video bitrate, number of interruptions, and number of quality switches (as discussed in Chapter 2.2). Besides, these application-layer objective QoE metrics can be effectively obtained directly either with an intermediate in network measurements tool or with a lightweight client-side instrumentation script.

Nevertheless, these metrics do not necessarily capture the severity of end-user viewing experience with respect to the underlying network conditions. For example, the average video bitrate does not sufficiently reflect the adaptation algorithm's efficiency in selecting the appropriate video bitrate for the segments with respect to either the underlying network conditions or available video bitrates. Similarly, the number of bitrate switches for a video session does not sufficiently reflect the intensity of the switching events. Since abruptly switching from a higher video bitrate to a lower bitrate can be easily perceived by the end-user, the amplitude of the bitrate switching event is a crucial QoE influence factor. In light of the above, we present a set of objective QoE metrics and use them for the objective QoE assessment. These metrics are adapted from the widely accepted objective video quality metrics reported in the literature [8, 21]. In addition, these objective QoE metrics can help the service providers to improve the video QoE and obtain good streaming performance.

Before defining the set of objective QoE metrics, we briefly review the basic notations used here. Let the k^{th} ($k = \{1, ..., K\}$) video segment downloaded by the client be of the video bitrate R_k and the quality level L_k. Let $max_j R_j$ denote the bitrate of the highest available representation, associated with the highest quality level, from available video adaptation set $\mathbb{R}$. Let $B(t)$ denote the buffer occupancy at time t. Let BW_T denote the bottleneck bandwidth, taken as the mean value of the bottleneck bandwidth estimated by the client across the download of K segments. Note that we used a single client in all the experiments. However, in the case of multiple streaming clients sharing a common link, the bottleneck bandwidth can be taken as the sum of the average bandwidths observed by the individual clients. For completeness, we summarize the notations used in defining the objective QoE metrics in Table 3.5. In what follows, we define the metrics used in the objective video quality evaluation.

1) **Bitrate Adaptation Efficiency (BAE)**: We can compute the average per-chunk video bitrate (R^{ψ}) over all the segments downloaded using Eq. (3.1). Nevertheless, a

Table 3.5: Summary of notations used in defining the objective QoE metrics.

Symbol	Description
K	total number of video segment downloaded by the client.
$\mathbb{IR}$	available video adaptation set that includes all the video bitrate levels.
R_k	video bitrate of the k^{th} ($k = \{1, ..., K\}$) video segment downloaded.
L_k	quality level of the k^{th} video segment downloaded.
$max_j R_j$	video bitrate of the highest available quality level from $\mathbb{IR}$.
$B(t)$	playback buffer occupancy at time t (represented as discrete instants of observation in seconds)
BW_T	bottleneck bandwidth computed by the client across the K segments.
t_s	start time of an interruption.
t_e	end time of an interruption.
ΔT	total duration of the video session (including both play and pause time).
ΔT_{Int}	total time the video playback is paused due to interruptions.

difficulty of simply using the average video bitrate is that it does not sufficiently reflect the adaptation algorithm's efficiency. Hence, leaving an open question about the adaptation algorithm performance under time-varying network conditions. Therefore, we define a metric to consider the average video bitrate played with respect to the bottleneck bandwidth and the highest available representation bitrate ($max_j R_j$). BAE is formally defined in Eq. (3.2).

$$R^{\psi} = \frac{1}{\mathrm{K}} \left(\sum_{k=1}^{K} R_k \right). \tag{3.1}$$

$$BAE = min \left(\frac{R^{\psi}}{min\left(max_j R_j, BW_T\right)}, \ 1 \right). \tag{3.2}$$

2) Interruption Ratio (IR): Playback interruptions are known to negatively impact the end-user viewing experience during a video streaming session. Two parameters play a vital role in deciding the video playback smoothness: interruption ratio and the average interruption duration. IR $\in$ $[0, 1]$ is defined as the ratio of the number of interruptions recorded and the number of segments downloaded during a streaming session by the client. Besides, IR estimates the severity of interruptions for the entire streaming session (i.e., based on the video length) instead of just counting the number of interruptions. A larger IR value denotes that the viewer suffers from frequent interruptions and could generally lead to significant QoE degradation. IR is formally

defined as:

$$IR = \frac{1}{K} \times \sum_{t=1}^{\Delta T} I\left(\{B(t-1) \neq 0\} \wedge \{B(t) = 0\}\right), \tag{3.3}$$

where $I(.)$ is the identity function defined at the time of interruption.

3) **Average Interruption Duration (AID)**: The general agreement is that we should avoid interruptions at all costs. However, there exists some disagreement among researchers when it comes to the acceptable duration of the interruption. In general, interruptions of shorter durations (e.g., of 0.25 seconds) are not noticeable by the end-user and do not lead to annoyance. This metric captures the average time for which the playback has been interrupted during the streaming session. Alternately, this measures the average time for which the buffer was empty. AID is formally defined as:

$$\begin{gathered} AID = \tfrac{1}{(K \times IR)} \times \left(\textstyle\sum_{j=1}^{(K \times IR)} |t_e - t_s|\right), \\ \text{where} \quad \forall j, \\ \{t_e > 0 \mid B(t_e) \neq 0 \wedge B(t_e - 1) = 0\} \\ \wedge \\ \{t_s > 0 \mid B(t_s) = 0 \wedge B(t_s - 1) \neq 0\}. \end{gathered} \tag{3.4}$$

4) **Bitrate Switching Amplitude Reward (BSAR)**: Bitrate switching amplitude refers to the difference between the representation levels for two subsequent video segments and is a crucial QoE influence factor. In general higher the switching amplitude, the worse the QoE impression for the end-user. However, the switching amplitude will be lower if the intense switches (i.e., from highest to lowest quality level or vice versa) have refrained during the streaming session. We define bitrate switching amplitude reward as a linear function of three components: (i) mean value of video segment quality, (ii) variation in video quality levels, and (iii) probability of avoiding bitrate switching events. The mean of video segment quality accounts for the benefit of selecting higher video quality levels. The variation in video quality levels captures the penalty due to frequent bitrate fluctuation across segments. This function ensures a high reward for selecting higher representations, staying among higher levels, and

minimizing the number of bitrate switches (particularly intense switches). Thus, it measures the overall impact of bitrate switching events in a streaming session. We formally define the reward function as:

$$BSAR = \left(\frac{\sum_{k=2}^{K} L_k}{(K-1)\times|\mathbb{R}|} + \frac{\sum_{k=2}^{K}(L_k - L_{k-1})}{|\mathbb{R}|} + \left(1 - \frac{\sum_{k=2}^{K} I(L_k)}{(K-1)}\right)\right), \tag{3.5}$$

where $I(L_k)$ is the identity function defined such that $I(L_k) = 1$ if and only if $L_k \neq L_{k-1}$, else it is zero.

5) **Video Continuity Index (VCI)**: The interruption ratio and the average interruption duration contribute the most in resulting in a non-pleasant viewing experience for the end-user. To this end, we define another metric video continuity index (VCI) to estimate the extent to which the interruptions are avoided by an adaptation algorithm. VCI $\in$ $[0, 1]$ is formally defined as:

$$VCI = \left(1 - \frac{\Delta T_{Int}}{\Delta T}\right). \tag{3.6}$$

6) **Initial Startup Delay Reward (ISDR)**: Initial startup delay (also known as startup delay) is generally present in all streaming applications. The initial startup delay (ISD) is the time spent by the client since sending the initial segment request for the stream before the playback starts. The Initial startup delay is unavoidable since the streaming client should download a certain number of video segments to build the buffer reservoir and minimize interruption-related impairments before starting the playback. It has been observed that starting the playback at the earliest (i.e., shorter startup delay) always maximizes the user experience and prevents the viewer from quitting the video session at an early stage [8]. The initial startup delay reward metric ensures that lower startup delay is rewarded while evaluating the QoE. The ISDR $\in$ $[0, 1]$ is formally defined as:

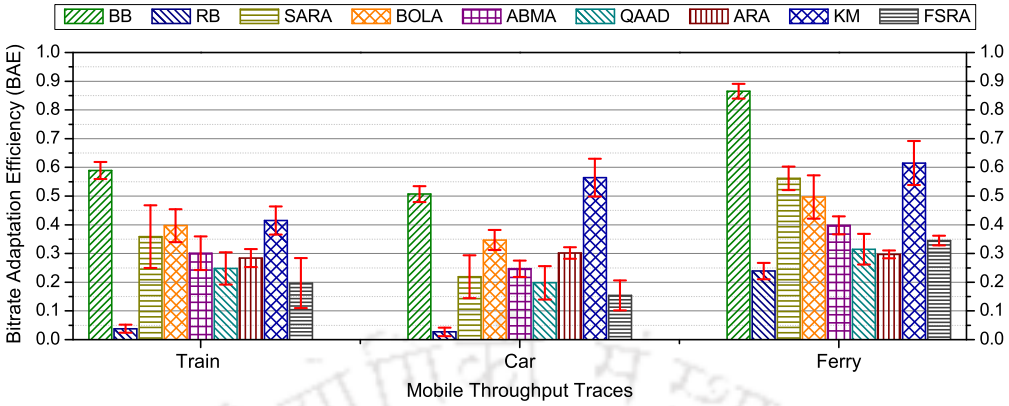
$$ISDR = \left(1 - \frac{ISD}{ISD^{MAX}}\right), \tag{3.7}$$

where ISD^{MAX} denotes the maximum startup delay observed across all the experiments for a given scenario.

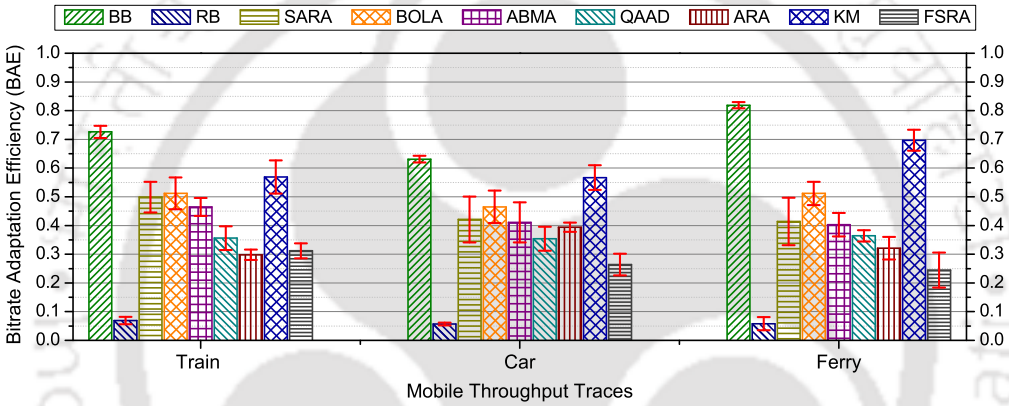
3.5.2 Quantitative Objective Assessment Results and Discussion

In this section, we report the results from the objective QoE assessment of DASH bitrate adaptation algorithms. We played three video sequences on a mobile phone with the three bandwidth profiles. The bandwidth profiles reflect both short-term and long-term bandwidth fluctuations in a cellular network when the user is moving. We measured the objective QoE metrics reported in Section 3.5.1 and considered the measurements to assess the QoE of the nine bitrate adaptation algorithms (discussed in Section 3.3).

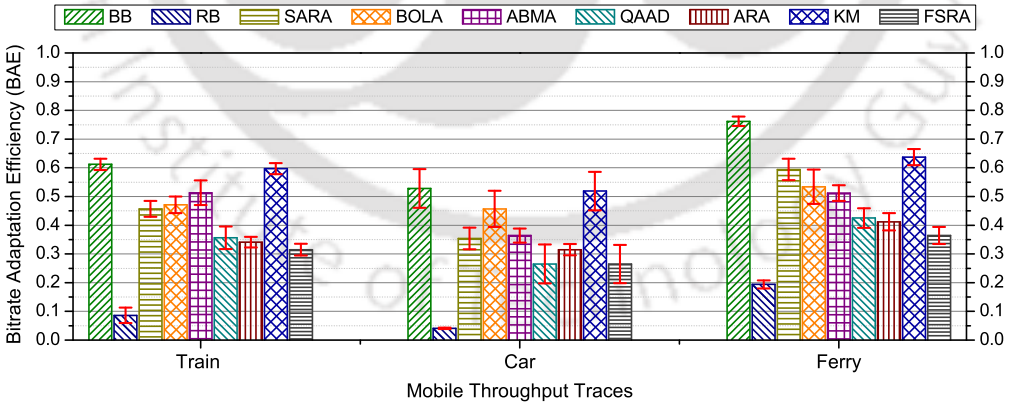
Fig. 3.8 shows the bitrate adaptation efficiency observed for the nine algorithms when streamed using the three traces and three video combinations. Fig. 3.8a shows the bitrate adaptation efficiency perceived by clients for the nine bitrate adaptation algorithms when the BBB video is streamed across the three traces. It is observed that the BB algorithm has the highest BAE across all three trace scenarios because the bitrate selection in BB is purely based on the current buffer occupancy. As the buffer gets filled, BB aggressively selects higher representation for fetching the segments without considering the impact on other parameters like playback interruptions and bitrate switching. Next, it is observed that the KM algorithm also records significantly higher BAE for all three traces. The higher BAE for the KM algorithm is attributed to the overestimation of the network bandwidth and reacting to the variation aggressively without considering the impact on other influencing factors. On the contrary, the RB algorithm performs poorly under all the scenarios due to the use of instantaneous bandwidth measurement for segment bitrate selection. We have observed that the RB algorithm fails to estimate the network bandwidth accurately, which is based on the historical network bandwidth estimation technique at the end of each segment download. The RB algorithm has ended up fetching video segments of lower bitrate only due to the inappropriate estimation. The video segments of lower bitrate have resulted in a significantly lower BAE value for RB as compared to the other algorithms across all the scenarios considered (as shown in Fig. 3.8a). It is also seen from Fig. 3.8a that SARA, BOLA, and ABMA+ have recorded moderate BAE values across all the trace-video scenarios. The SARA follows a conservative approach in bitrate selection, leveraging the time required to download the future segment and stabilize the buffer occupancy to avoid unnecessary interruptions. Due to this, it has a moderate BAE across all the scenarios. The BOLA also follows a conservative approach in segment bitrate selection by merely utilizing



(a) Big Buck Bunny (BBB)



(b) Red Bull Playstreets (RBPS)



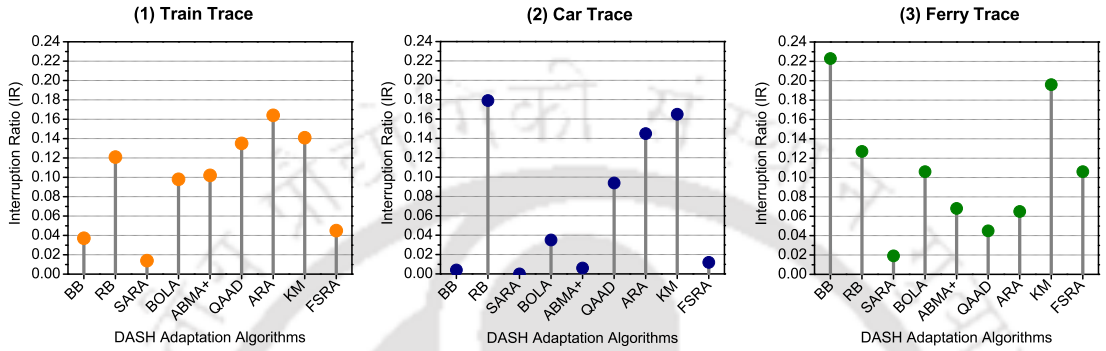
(c) Tears of Steel (ToS)

Figure 3.8: Performance statistics of bitrate adaptation efficiency for different trace-algorithm combinations when streamed with: (a) Big Buck Bunny (b) Red Bull Playstreets and (c) Tears of Steel.

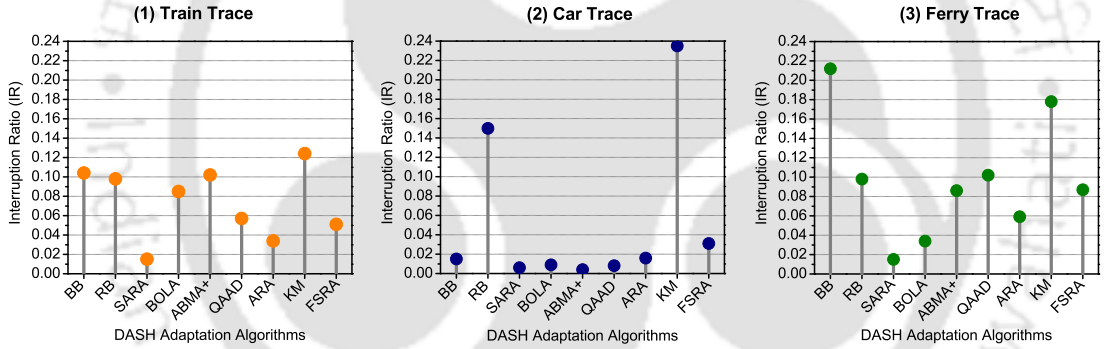
the current buffer occupancy. The moderate BAE value recoded by BOLA is attributed to the function that makes the control decisions with a goal to maximize the playback utility. It is also evident from Fig. 3.8a that all the adaptation algorithms have recorded a significantly higher BAE when streamed with ferry trace. This is mostly due to the availability of higher bandwidth in the ferry trace profile at the beginning of the video session. We observe similar performance for all the nine bitrate adaptation algorithms across the three trace profiles when streamed with the RBPS (as shown in Fig. 3.8b) and ToS (as shown in Fig. 3.8c) video sequences.

Fig. 3.9 shows the interruption ratio observed for the nine algorithms across different experimental scenarios. Fig. 3.9a, Fig. 3.9b, and Fig. 3.9c depicts the IR observed for the algorithms when streamed with Big Buck Bunny, Red Bull Playstreets, and Tears of Steel video sequences, respectively. From Fig. 3.9a, we can observe that SARA leads to very few interruptions (lower IR) since it considers segment size in bandwidth estimation and it avoids buffer underflows leveraging the buffer thresholds. On the other hand, we observe a higher IR for the RB and KM algorithms across all three trace profiles. It is also evident from the values reported in Table 3.6 that RB has interruptions of smaller durations (resulting in a lower average interruption duration) under the three trace scenarios due to buffer depletion before receiving the segment. We also observe from Table 3.6 that KM has recorded interruptions of sufficiently longer durations (resulting in a higher AID). To understand the reasons behind these results, we refer to the higher BAE values recorded by KM (as depicted in Fig. 3.8) with an aggressive approach for segment bitrate selection. We observe that the selection of higher video bitrate and inappropriate bandwidth estimation by the KM algorithm has resulted in frequent playback interruptions of longer duration. From Fig. 3.9, we observe that both BOLA and ABMA+, which follow a conservative approach in bitrate selection, have also recorded moderate IR under train and ferry trace scenarios. It is also evident from Fig. 3.9 that the majority of the algorithms have recorded significantly lower IR under the car trace profile. The lower IR values and lower AID values (refer to Table 3.6) observed for the algorithms (except RB) is mostly attributed to the higher bandwidth available in the car trace profile together with a significantly lower number of bandwidth fluctuations. On the contrary, the higher IR (shown in Fig. 3.9) and AID values observed for RB under the car trace are attributed to the instantaneous bandwidth

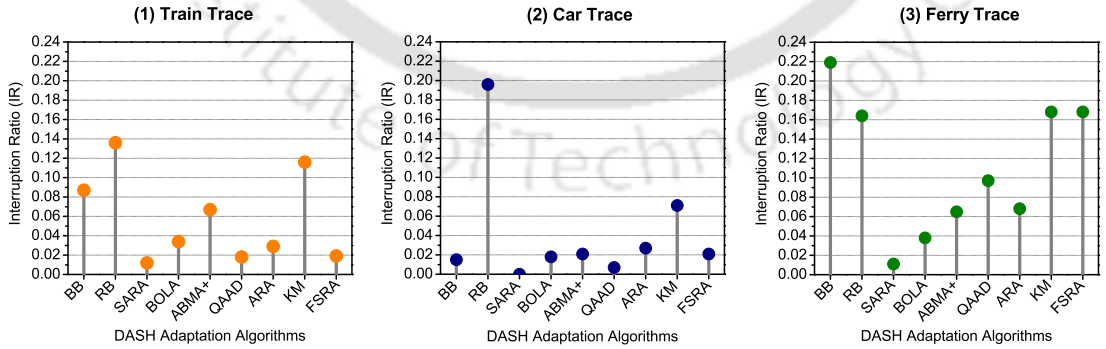
estimation technique employed by the algorithm. The bandwidth estimation technique in RB not only estimates the bandwidth inappropriately but also does not maintain a buffer reservoir. Thus, it leads to frequent playback depletions due to delayed segment arrival.



(a) Big Buck Bunny (BBB)



(b) Red Bull Playstreets (RBPS)



(c) Tears of Steel (ToS)

Figure 3.9: Performance statistics of interruption ratio for different trace-algorithm combinations when streamed with: (a) Big Buck Bunny (b) Red Bull Playstreets and (c) Tears of Steel.

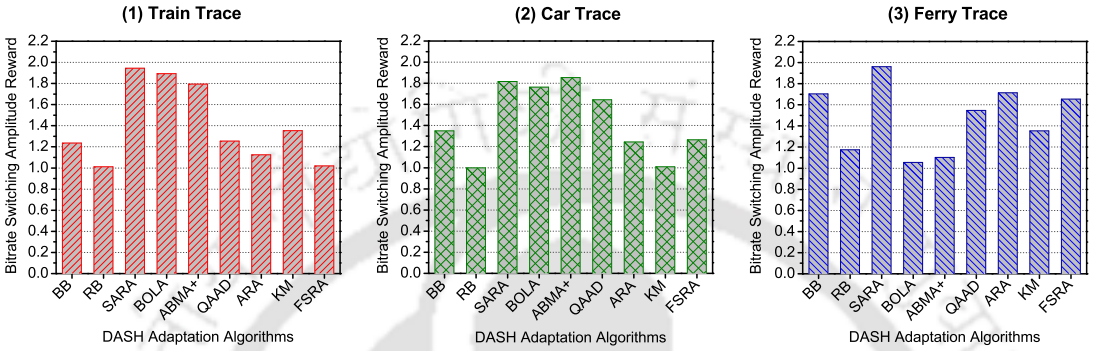
In the ferry trace scenario, we see from Fig. 3.9 that the BB algorithm has the highest IR and also a longer average interruption duration across the three video sequences. The higher IR observed for BB is mostly attributed to the bandwidth variations in the ferry trace profile. Although the BB algorithm selects a higher bitrate for segment(s) based on the current buffer occupancy, a sharp drop of bandwidth in the ferry trace profile causes buffer depletions for a longer time duration. We have observed similar patterns for the BB algorithm when streamed with the three video sequences (see Fig. 3.9).

Table 3.6: Measured objective QoE metrics when streamed with the Big Buck Bunny video sequence.

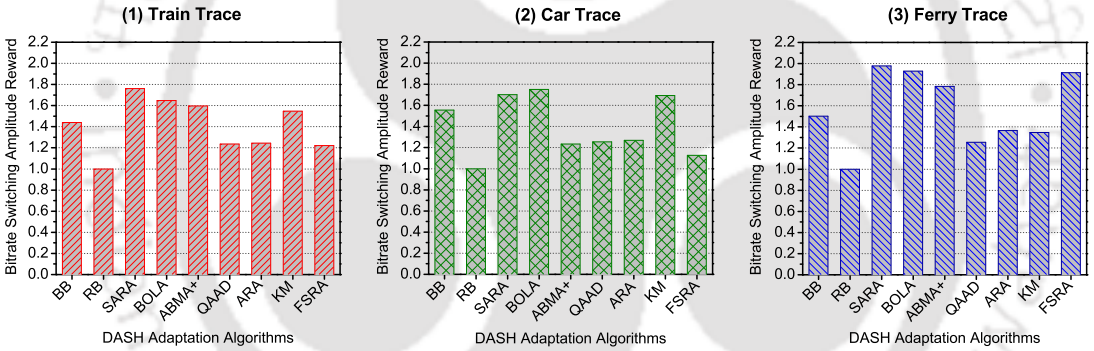
Trace	Adaptation Algorithms	Big Buck Bunny					
		BAE	IR	AID	BSAR	VCI	ISDR
Train	BB	0.589	0.037	10.0640	1.235	0.9641	0.418
	RB	0.038	0.121	4.57004	1.012	0.9476	0.451
	SARA	0.359	0.014	6.05083	1.945	0.9916	0.431
	BOLA	0.397	0.098	7.69229	1.894	0.9299	0.435
	ABMA+	0.301	0.102	6.33555	1.794	0.9393	0.498
	QAAD	0.248	0.135	12.0986	1.254	0.8596	0.501
	ARA	0.284	0.164	4.45588	1.124	0.9319	0.495
	KM	0.415	0.141	9.26115	1.354	0.8845	0.397
	FSRA	0.197	0.045	2.13136	1.019	0.9905	0.458
Car	BB	0.507	0.004	15.5967	1.349	0.9938	0.492
	RB	0.027	0.179	3.57223	1.000	0.9399	0.501
	SARA	0.219	0.000	0.00000	1.817	1.0000	0.518
	BOLA	0.347	0.035	2.42033	1.764	0.9916	0.548
	ABMA+	0.247	0.006	0.33340	1.854	0.9998	0.497
	QAAD	0.198	0.094	1.22675	1.644	0.9886	0.467
	ARA	0.302	0.145	1.22839	1.245	0.9825	0.398
	KM	0.564	0.165	4.45681	1.009	0.9315	0.514
	FSRA	0.154	0.012	3.26272	1.265	0.9961	0.535
Ferry	BB	0.865	0.223	7.24552	1.703	0.8609	0.337
	RB	0.239	0.127	6.10915	1.174	0.9280	0.561
	SARA	0.562	0.019	5.15527	1.962	0.9903	0.511
	BOLA	0.497	0.106	7.33041	1.054	0.9279	0.378
	ABMA+	0.398	0.068	3.98767	1.103	0.9736	0.488
	QAAD	0.315	0.045	4.55829	1.546	0.9799	0.542
	ARA	0.297	0.065	3.18779	1.714	0.9797	0.339
	KM	0.615	0.196	12.9791	1.354	0.7972	0.452
	FSRA	0.345	0.106	6.32152	1.654	0.9372	0.491

We compute the bitrate switching amplitude reward for all the algorithms across the experimental scenarios considered. Fig. 3.10 presents the BSAR values computed for

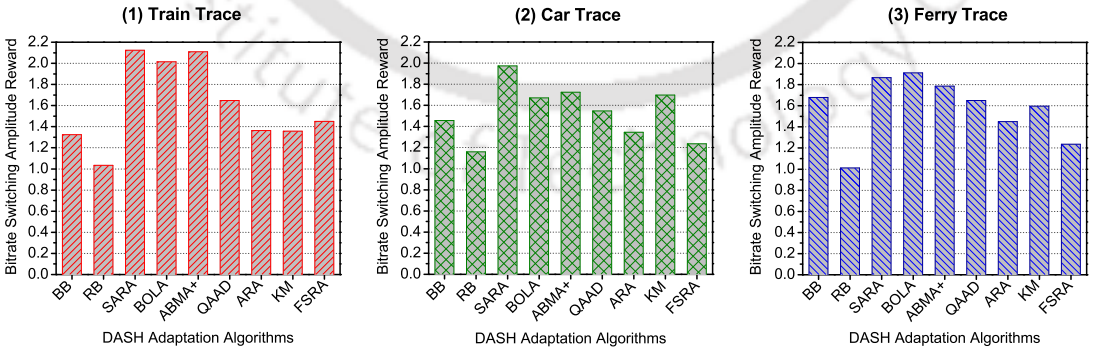
all the algorithms across the trace-video combinations. From Fig. 3.10, we observe that SARA, BOLA and ABMA+ have recorded higher BSAR under all the trace-video scenarios considered. In the train trace scenario, we see from Fig. 3.10 that SARA has slightly



(a) Big Buck Bunny (BBB)



(b) Red Bull Playstreets (RBPS)



(c) Tears of Steel (ToS)

Figure 3.10: Performance statistics of bitrate switching amplitude reward for different trace-algorithm combinations when streamed with: (a) Big Buck Bunny (b) Red Bull Playstreets and (c) Tears of Steel.

Table 3.7: Measured objective QoE metrics when streamed with the Red Bull Playstreets video sequence.

Trace	Adaptation Algorithms	Red Bull Playstreets					
		BAE	IR	AID	BSAR	VCI	ISDR
Train	BB	0.726	0.104	12.6421	1.439	0.8838	0.159
	RB	0.069	0.098	2.25274	1.000	0.9784	0.178
	SARA	0.499	0.015	1.80487	1.762	0.9973	0.239
	BOLA	0.512	0.085	3.05373	1.648	0.9747	0.165
	ABMA+	0.465	0.102	6.00325	1.597	0.9423	0.197
	QAAD	0.356	0.057	2.59935	1.235	0.9854	0.264
	ARA	0.298	0.034	2.61121	1.245	0.9912	0.241
	KM	0.569	0.124	8.94066	1.546	0.9002	0.197
	FSRA	0.312	0.051	2.02061	1.221	0.9898	0.213
Car	BB	0.631	0.015	5.30860	1.556	0.9921	0.036
	RB	0.057	0.150	1.10015	1.000	0.9985	0.181
	SARA	0.421	0.006	0.18956	1.701	0.9963	0.174
	BOLA	0.465	0.009	1.00090	1.751	0.9991	0.219
	ABMA+	0.411	0.004	0.75022	1.234	0.9997	0.246
	QAAD	0.354	0.008	1.00080	1.254	0.9992	0.175
	ARA	0.394	0.016	1.00160	1.268	0.9984	0.168
	KM	0.567	0.240	1.28423	1.694	0.9701	0.211
	FSRA	0.264	0.031	0.61406	1.126	0.9981	0.197
Ferry	BB	0.819	0.212	12.8503	1.501	0.7859	0.000
	RB	0.058	0.098	3.58056	1.000	0.9661	0.313
	SARA	0.414	0.015	4.56437	1.977	0.9932	0.439
	BOLA	0.512	0.034	5.14736	1.928	0.9828	0.364
	ABMA+	0.403	0.086	2.95706	1.784	0.9752	0.248
	QAAD	0.364	0.102	2.30798	1.254	0.9770	0.248
	ARA	0.321	0.059	1.62561	1.365	0.9905	0.194
	KM	0.697	0.178	10.8365	1.348	0.8383	0.267
	FSRA	0.245	0.087	2.11881	1.914	0.9819	0.215

outperformed compared to others, especially both BOLA and ABMA+ in terms of BSAR. The slight improvement in BSAR is mostly because the SARA algorithm avoids frequent bitrate switches or switches of higher amplitude by maintaining a stable buffer reservoir and avoids buffer underflows (leading to interruptions). From Fig. 3.10, it can be seen that the RB fares poorly in terms of BSAR in all the experimental scenarios. Although the RB avoids bitrate switches with a higher amplitude, The lower BSAR value for RB is mainly attributed to the fact that it fetches segments with lower video bitrate only (as evident from Fig. 3.8). It can be seen from Fig. 3.10 that all other algorithms that include BB, QAAD, ARA and FSRA have recorded a moderate BSRA for all the trace-video scenarios.

Table 3.8: Measured objective QoE metrics when streamed with the Tears of Steel video sequence.

Trace	Adaptation Algorithms	Tears of Steel					
		BAE	IR	AID	BSAR	VCI	ISDR
Train	BB	0.612	0.087	6.98653	1.324	0.9427	0.426
	RB	0.086	0.136	4.56874	1.034	0.9415	0.467
	SARA	0.457	0.012	1.67001	2.124	0.9980	0.516
	BOLA	0.471	0.034	2.61121	2.014	0.9912	0.458
	ABMA+	0.513	0.067	3.20156	2.109	0.9790	0.398
	QAAD	0.356	0.018	1.67168	1.648	0.9970	0.452
	ARA	0.341	0.029	1.28060	1.364	0.9963	0.415
	KM	0.597	0.116	4.27053	1.357	0.9528	0.511
	FSRA	0.315	0.019	1.90158	1.451	0.9964	0.454
Car	BB	0.528	0.015	2.00601	1.457	0.9970	0.484
	RB	0.041	0.196	4.37634	1.159	0.9210	0.546
	SARA	0.354	0.000	0.00000	1.974	1.0000	0.549
	BOLA	0.457	0.018	0.27791	1.671	0.9995	0.512
	ABMA+	0.364	0.021	0.95428	1.724	0.9980	0.534
	QAAD	0.265	0.007	1.71635	1.546	0.9988	0.498
	ARA	0.315	0.027	1.30085	1.345	0.9965	0.434
	KM	0.519	0.071	1.97086	1.697	0.9862	0.481
	FSRA	0.265	0.021	0.95429	1.235	0.9980	0.502
Ferry	BB	0.762	0.219	13.6008	1.679	0.7705	0.461
	RB	0.194	0.164	5.63528	1.012	0.9154	0.358
	SARA	0.594	0.011	2.46119	1.867	0.9973	0.493
	BOLA	0.534	0.038	5.04216	1.912	0.9812	0.511
	ABMA+	0.512	0.065	3.88006	1.786	0.9754	0.497
	QAAD	0.425	0.097	2.60004	1.651	0.9754	0.425
	ARA	0.412	0.068	4.03418	1.451	0.9733	0.386
	KM	0.637	0.168	13.1370	1.597	0.8192	0.461
	FSRA	0.364	0.168	0.13722	1.235	0.9977	0.358

We observe from Table 3.6 that all the nine algorithms across the three trace profiles have recoded similar ISDR. The similar ISDR value for all the algorithms is mainly because all of them fetch segments of lower bitrate at the beginning of the video session to build the buffer reservoir. Across all the three video scenarios, we have observed that BB has recorded significantly lower ISDR. This is mostly because BB fetches segments and builds a sufficient buffer reservoir (in terms of segments) before starting the video session. For completeness, we have presented the values recorded during the objective QoE assessment with the three video sequences, namely BBB, RBPS and ToS in Table 3.6, Table 3.7 and Table 3.8, respectively. These values were subsequently used for computing a cumulative

QoE score for the nine algorithms (refer Section 3.6.4).

3.6 Linear Parametric QoE Scoring Model

From the discussion of objective quality assessment results, we observe mutual dependency among the objective QoE metrics. In addition, these objective QoE metrics are interrelated in complex counter-intuitive ways, and their relationship can be unpredictable under time-varying network conditions. To address the issue of inter-dependency among objective QoE metrics for assessing the performance bitrate adaptation algorithm, we propose a parametric QoE scoring model to compute a cumulative QoE score for a DASH session. In this section, we also detail the post-survey measures (collected at the end of the subjective experiments) about the end-users preferences and expectations for a DASH video session. Next, we compute the ranking/weights for the objective QoE metrics from the post-survey measures and consider them in the proposed parametric QoE scoring model.

3.6.1 Quantitative Post-survey Measures

At the end of each viewing session, the participants were asked to answer several questions about their viewing experience and expectations. The post-survey (also known as exit-survey) was used to collect feedback about the participants' viewing experience and preference/expectation for the objective QoE metrics in a quantitative manner. We have used a five-level Likert scale, a psychometric scale widely used to record responses in the survey. The five-level Likert scale helps the participants indicate their experience for interruptions and bitrate switching events (feeling of being imperceptible to very annoying), preference for video quality, and initial startup delay (from no opinion to very important). The participants were asked to choose an option that best summarizes their opinion about the objective metrics affecting QoE. We represent the values with suitable phrases that allow the test participants to comprehend the Likert scale in a simplified manner. The Likert scale with suitable phrases will avoid the risk of containing bias and negatively impacting the information collected during the survey. For recoding the participants viewing experience (i.e., opinion for interruptions and bitrate switching events), the Likert scale presented with simplified phrases: imperceptible (1), perceptible but not annoying, slightly annoying, annoying, and very annoying (5). Higher the annoyance, the larger the value it represents on

the Likert scale. Similarly, for recoding the participants' preference (i.e., opinion for video quality and initial startup delay), the Likert scale presented with a phrase-rating mapping: no opinion (1), not important, moderately important, important, and very important (5). Higher the viewer preference for the metric, the larger the value on the Likert scale. We summarize the numeric values used as part of the phrase-rating mapping (i.e., rating scale and their corresponding phrases) in Table 3.9.

Table 3.9: Summary of phrase-rating mapping considered for the post-survey.

Verbal Phrases	**Numeric Value**
Imperceptible / No Opinion	1
Perceptible but not Annoying / Not Important	2
Slightly Annoying / Moderately Important	3
Annoying / Important	4
Very Annoying / Very Important	5

Since the actual number of responses in the survey result varied for each phrase, the participant responses were converted into percentiles. The conversion responses to percentiles were performed by dividing the number of responses for a phrase by the total number of responses in the survey (i.e., eighty-seven). The percentage of responses recorded for each of the phrases based on the participant's experience with interruption and bitrate switching events are reported in Table 3.10. From Table 3.10, we see that about 28.7% of the participants felt that interruptions were "slightly annoying", while 16.09% felt that they were "very annoying". Similarly, the percentages computed for each of the phrases based on their preference with video bitrate quality and initial startup delay are reported in Table 3.11. We see that about 73% of the users gave high importance to the average bitrate played and about 50% of the users felt that the ISD affects their experience.

Table 3.10: Percentage of users providing ratings on a five-point scale reflecting their experience for interruptions and bitrate switching events (BSE).

Verbal Phrases (numeric value)	**Percentage of Users**	
	Interruptions	**BSE**
Imperceptible (1)	3.45	10.34
Perceptible but not Annoying (2)	19.54	21.84
Slightly Annoying (3)	28.74	44.83
Annoying (4)	32.18	17.24
Very Annoying (5)	16.09	5.75

Table 3.11: Percentage of users providing ratings on a five-point scale reflecting preference for average video bitrate and initial startup delay.

Verbal Phrases (numeric value)	Percentage of Users	
	Avg. bitrate	ISD
No Opinion (1)	2.30	3.45
Not Important (2)	5.75	21.84
Moderately Important (3)	18.39	25.29
Important (4)	49.43	39.08
Very Important (5)	24.14	10.34

3.6.2 Deriving Objective QoE Metrics Ranking

Next, we computed weights (parameters) corresponding to objective QoE metrics from the user ratings (collected during the post-survey), reflecting each metric's importance on the end-user viewing experience. For a QoE metric, $Numerical\_value(j)$ is the numerical value corresponds to the verbal phrase (as shown on Table 3.9) and P_j is the percentage of responses for respective verbal phrase (refer Table 3.10 and Table 3.11). The weight for a QoE metric is then obtained using the following formula:

$$Weight = \sum_{j} Numerical\_value(j) \times \left(\frac{P_j}{100}\right), \quad \forall j \in verbal\ phrase \tag{3.8}$$

With this approach, we compute the weights corresponding to BAE, BSRA, IR, and ISDR as $\lambda = 3.87$, $\mu = 2.86$, $\rho = 3.38$, and $\sigma = 3.31$, respectively. The larger the weight parameter value, the higher the corresponding metric's influence on the users' viewing experience. Subsequently, these weights are used in the parametric QoE scoring model to calculate the QoE score for the algorithms as explained in Section 3.6.3.

3.6.3 Proposed QoE Scoring Model

In general, any DASH streaming client is designed to minimize or maximize either a few or all of the QoE metrics to ensure a better viewing experience for end-users. From the research literature, it is still not clear as to how the user-perceived QoE can be measured quantitatively by these metrics. In a bid to design a quantitative measure for QoE, we propose a linear parametric QoE scoring model to compute a cumulative QoE score for a DASH session. Our proposed model can also help to assess the QoE with different algorithms

designed to optimize different metrics. The parametric QoE scoring model is similar to the one proposed in [68] but uses the objective QoE metrics defined in this Chapter (refer Section 3.5.1). We formally define the proposed model in Eq. (3.9). In Eq. (3.9), λ, μ, ρ, and σ are non-negative weights assigned to the objective QoE metrics. The weights in the proposed QoE scoring model are determined to be $\lambda = 3.87$, $\mu = 2.86$, $\rho = 3.38$, and $\sigma = 3.31$ in Section 3.6.2. In summary, our proposed parametric QoE scoring model to compute the QoE score is fairly simple and considers only notable objective QoE metrics. However, a parametric model like this can be extended to include other metrics or varying influencing QoE factors based on the users' preferences.

$$QoE_1^K = (\lambda \times BAE) + (\mu \times BSRF) + (\rho \times (1 - IR)) + (\sigma \times ISDR) + VCI. \quad (3.9)$$

3.6.4 QoE Score Estimations

In this subsection, we have computed the QoE score for all nine algorithms to assess their overall performance. We have computed the cumulative QoE score using the formula given in Eq. (3.9). The computed QoE score can only range from 0 to 20. The score of 0 is the lowest possible QoE score while 20 becomes the highest QoE score that can only be obtained using the formula given in Eq. (3.9). Note that the higher the QoE score better the performance of the bitrate adaptation algorithm in terms of overall QoE. In Table 3.12, we present the QoE score for all the algorithms evaluated across different scenarios, i.e., three video sequences and three trace profiles. Furthermore, we have highlighted the QoE score computed using linear parametric QoE scoring model for DASH adaptation algorithms while streaming with three different video sequences, namely Big Buck Bunny, Red Bull Playstreets and Tears of Steel in Fig. 3.11, Fig. 3.12 and Fig. 3.13, respectively.

From Table 3.12, we can observe that SARA has outperformed the other algorithms in terms of the QoE score in most of the scenarios considered (as highlighted in gray color). Although the SARA has recorded a moderate BAE, the higher QoE score is mainly due to lower IR and higher BSER values obtained for the SARA algorithm across all the scenarios considered. We have also observed that the BOLA has slightly outperformed SARA in terms of QoE score, especially with car trace profile and when streamed with BBB and RBPS video sequences. The slightly higher QoE score achieved by BOLA is attributed to the

higher BAE by the BOLA (as compared to SARA) under favorable bandwidth conditions (i.e., with car trace profile). It can be seen from Table 3.12 that ABMA+ with moderate BAE and lower IR has also recorded comparatively a higher QoE score for all the scenarios. On the contrary, RB performs poorly in terms of the QoE score in all the trace-video scenarios, as the individual metric-wise performance for RB is also poorer. In addition, the low QoE score for RB is mainly attributed to the lower BAE and higher IR observed across all the scenarios. From Table 3.12, it can be observed that all other algorithms that

Table 3.12: QoE score computed using linear parametric QoE scoring model for bitrate adaptation algorithms of DASH.

Video Sequence	Adaptation Algorithm	Estimated QoE Score*		
		Train	Car	Ferry
Big Buck Bunny	BB [76]	11.414 (0.4112)	11.809 (0.3574)	12.821 (0.2131)
	RB [71]	8.4528 (0.6115)	8.3376 (0.3323)	10.018 (0.4724)
	SARA [39]	12.703 (0.0914)	12.139 (0.2511)	13.784 (0.0917)
	BOLA [37]	12.372 (0.1376)	12.455 (0.1007)	10.139 (0.8278)
	ABMA+ [40]	11.918 (0.5686)	12.263 (0.3139)	10.434 (0.4679)
	QAAD [87]	9.9878 (0.1908)	11.065 (0.3173)	11.642 (0.9618)
	ARA [80]	9.7097 (0.3915)	9.9192 (0.1852)	11.313 (0.5141)
	KM [79]	10.580 (1.2236)	10.524 (0.9369)	11.263 (0.3065)
	FSRA [74]	9.4111 (0.3703)	10.320 (0.4317)	11.649 (0.4741)
Red Bull Playstreets	BB [76]	11.364 (0.5552)	11.332 (0.4126)	10.915 (0.4608)
	RB [71]	7.7434 (0.3611)	7.5512 (0.4978)	8.1354 (0.2581)
	SARA [39]	12.088 (0.1626)	11.426 (0.0626)	13.032 (0.0521)
	BOLA [37]	11.308 (0.3156)	11.881 (0.1206)	12.948 (0.0891)
	ABMA+ [40]	10.996 (0.2733)	10.300 (0.3701)	11.547 (0.2625)
	QAAD [87]	9.9564 (0.3995)	9.8878 (0.5492)	9.8282 (0.7127)
	ARA [80]	9.7679 (0.1937)	10.032 (0.2747)	9.9594 (0.2141)
	KM [79]	11.137 (1.0445)	11.276 (0.7622)	11.053 (0.8927)
	FSRA [74]	9.6019 (0.3698)	9.1674 (0.2201)	11.202 (0.7246)
Tears of Steel	BB [76]	11.594 (0.3412)	12.139 (0.1521)	12.687 (0.1124)
	RB [71]	8.6976 (0.3102)	8.9191 (0.1692)	8.5711 (0.4634)
	SARA [39]	13.889 (0.0591)	13.213 (0.2062)	13.610 (0.0569)
	BOLA [37]	13.355 (0.0331)	12.561 (0.1396)	13.459 (0.0484)
	ABMA+ [40]	13.467 (0.0592)	12.414 (0.1318)	12.870 (0.2717)
	QAAD [87]	11.903 (0.4199)	11.451 (0.3697)	11.801 (0.2907)
	ARA [80]	10.873 (0.1415)	10.787 (0.2518)	11.145 (0.3012)
	KM [79]	11.824 (0.7105)	12.580 (0.2637)	12.189 (0.6591)
	FSRA [74]	11.184 (0.6708)	10.526 (1.0554)	9.9356 (0.9042)

* Values represent the mean from multiple measurements with confidence interval in brackets for all the trace-algorithm combinations.

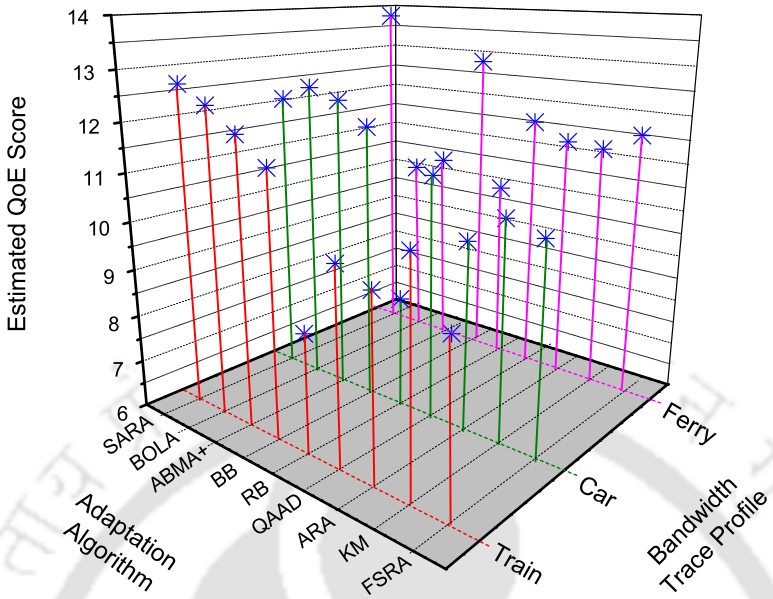


Figure 3.11: Illustration of linear parametric QoE score for DASH adaptation algorithms while streaming with Big Buck Bunny video sequence.

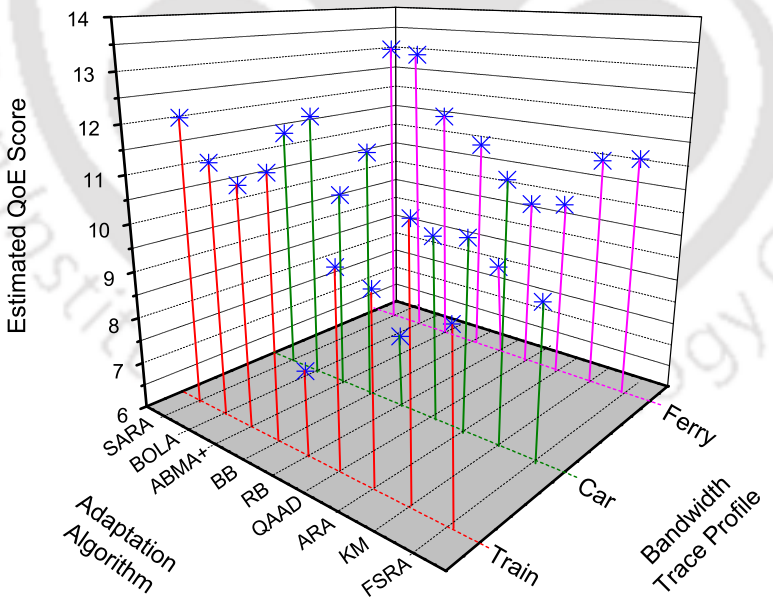


Figure 3.12: Illustration of linear parametric QoE score for DASH adaptation algorithms while streaming with Red Bull Playstreets video sequence.

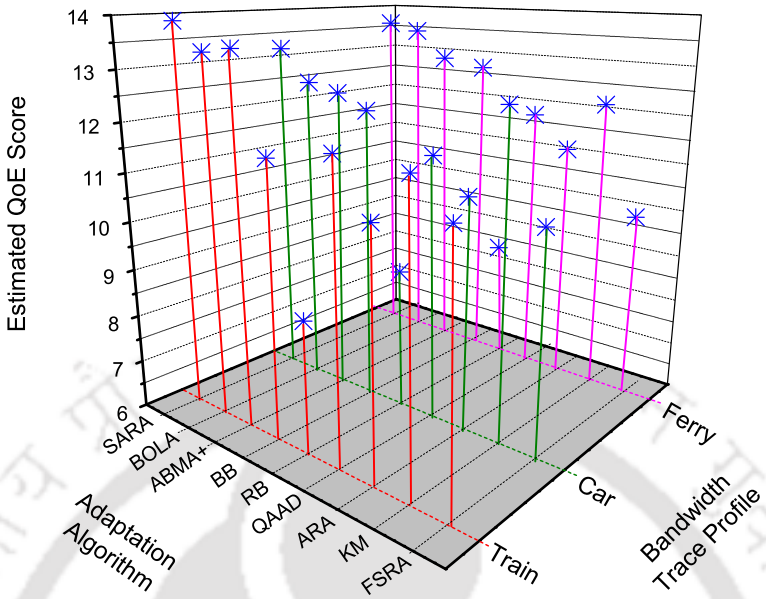


Figure 3.13: Illustration of linear parametric QoE score for DASH adaptation algorithms while streaming with Tears of Steel video sequence.

include ARA, QAAD, FSRA, BB and KM have recorded moderate QoE scores. The BB and KM algorithms record a moderate QoE score mostly due to the higher BAE (as a result of aggressively fetching segments with a higher bitrate) at the cost of several playback interruptions (leads to a higher IR). The QoE score uses the weights derived from user feedback that captures the user's perception of the objective metrics' importance. It also captures the algorithms' performance in terms of individual objective metrics and helps to compare the performance of different bitrate adaptation algorithms whose objectives vary in optimizing the individual metrics.

3.7 Summary

In this chapter, we performed a video QoE assessment of DASH bitrate adaptation algorithms using a mobile trace-driven emulation testbed. We performed both subjective and objective assessments of the bitrate adaptation algorithms in mobile devices. From the subjective QoE assessment, we observed that the algorithm considering both throughput and buffer occupancy performs better in terms of MOS. In contrast, the algorithm that merely considers

only estimated network throughput for segment bitrate selection exhibits poor MOS during the subjective assessment. For objective QoE assessment, we have formally defined a set of six objective QoE metrics to capture the severity of the adaptation algorithm performance. From the objective QoE assessment, we found that the RB approach, which lacks in estimating the network bandwidth, fares poorly in terms of the objective metrics. In particular, the RB records a lower BAE and higher IR mostly due to the inappropriate bandwidth estimation. Our objective QoE assessment also revealed that the BB exhibits higher BAE compared to the other algorithms, despite having significantly higher IR and longer AID. However, the algorithms that consider both estimated throughput (not using the history-based estimation technique) and current buffer occupancy fares with moderate BAE and lower IR. Among these algorithms, SARA outperforms all others in terms of all the objective QoE metrics, especially with moderate BAE and significantly lower IR. We have also computed a cumulative QoE score based on the proposed linear parametric scoring model. The QoE score obtained can be used to compare the performance of different bitrate adaptation algorithms whose end-goal varies in optimizing the individual QoE metrics. Overall, it is observed that bitrate adaptation algorithms that make use of both throughput and buffer information and follow a conservative approach in selecting the video bitrate level for future segments fare well in terms of the QoE. In the next chapter, we propose a QoE prediction model based on multinomial logistic regression.

CHAPTER

4

Predicting Quality of Experience in DASH Using Parametric Multinomial Logistic Regression Model

The previous chapter focused on video QoE assessment that includes both subjective assessment and objective assessment of popular DASH bitrate adaptation algorithms in mobile devices. The chapter particularly provides a detailed discussion on how the bitrate adaptation algorithms fare in mobile devices operating under highly variable bandwidth conditions. It also presents how the client's quality decision process, as it determines the actual video quality to be requested for the segments, negatively influences QoE metrics such as playback interruptions and bitrate switches based on the network conditions.

In light of the wide adoption of the HAS, the market for video service providers is highly competitive and fast-growing. The evolving world of video streaming not only offers opportunities to the video service providers but also poses several challenges, such as providing a satisfactory viewing experience to the end-users. Accurately predicting/estimating the QoE as a single score is paramount for the service providers to improve the service delivery process. To solve this problem, this chapter presents a parametric quality-estimation model based on multinomial logistic regression to predict the overall QoE score for a HAS video session. The proposed model is capable of predicting both short-term and long-term QoE scores by merely taking a set of objective QoE metrics as input. In this chapter, the design of the proposed multinomial logistic regression-based model is detailed and validated the prediction accuracy of the model using the k-fold cross-validation technique. The proposed model particularly shows a better prediction of QoE with a reasonably low

computational complexity, suggesting its suitability for online use when compared with three other benchmarking models in the literature.

Organization of the Chapter: The remainder of this chapter is structured as follows. We first present the motivation for developing a new parametric QoE-estimation model in Section 4.1. Section 4.2 details the proposed parametric QoE-estimation model based on multinomial logistic regression. In Section 4.3, we present the proposed QoE model performance evaluation that includes the validation of prediction accuracy, deployment complexity compared to existing models, prediction of quality scores for different bitrate adaptation algorithms, and model applicability for longer duration DASH videos. Finally, we conclude the chapter in Section 4.4.

4.1 Motivation

The mean opinion score (MOS) is the most popular and the de facto metric for subjective QoE assessment. Although MOS provides a better understanding of video quality experienced by the end-users, the subjective QoE assessment has inherent drawbacks. Subjective assessment incurs both cost and time and impractical for real-world applications as obtaining the MOS feedback directly from the viewer is not easy. Furthermore, conducting large-scale MOS measurements to obtain the subjective perception that is statistically meaningful is an onerous job.

In recent years, as an alternative solution to the subjective assessment approach, researchers have envisaged QoE models to predict/estimate the overall viewing experience. In practice, the QoE prediction models, especially for mobile video streaming, can be used by a variety of stakeholders like mobile network operators, video streaming service/platform providers, and regulators. For example, network operators, service/platform providers, and regulators can use the QoE prediction model(s) concomitantly to improve network configuration, proactively address the end-user issues suffering from poor QoE, and determine the universal service requirements[1] intending to ensure an adequate QoE for streaming services, respectively.

[1]India has a universal service obligation of a 2 Mbps broadband-on-demand connections to 600 million citizens by the year 2020.

The QoE prediction models in the literature, based on the type of input and the location where the input information is extracted, can be broadly classified into four categories: (i) media-layer models, (ii) bitstream models, (iii) parametric models, and (iv) hybrid models. A detailed discussion of the QoE model classification for streaming applications is presented in Chapter 2.5.2. Firstly, the media-layer QoE models use complex perceptually based psychophysical models to exploit the actual media signal information and estimate the video quality. The PSNR [103] and SSIM [104], VMAF [105], VQM [106], and MVQM [107] have been some of the widely used full reference media-layer models to estimate video QoE. However, these signal-based models have to deal with the non-trivial problem of synchronizing the source input signal to the decoded output signal at the end-user equipment. In addition, the media-layer models have higher computation complexity for long video sequences with several video bitrate levels. Secondly, bitstream models utilize the encoded bitstream and packet layer information to estimate the video QoE and have received wider acceptance for adaptive multimedia streaming services. Despite being relatively computationally inexpensive, the bitstream models are ill-suited for real-time deployment because they are only suitable for the specific codec(s). Next, the parametric QoE models either consider network-related factors (including various QoS parameters) or numerous application-level metrics (such as objective QoE metrics) for predicting the video quality score, which closely resembles the perceived video quality. Nonetheless, the quality prediction by packet-layer-based parametric QoE models (which utilizes the information extracted from packet headers) in the long-run does not necessarily correlate with the end-user perceived quality, mostly due to compression and/or packet losses during transmission [8].

The authors in [21] proposed a set of industry-standard objective QoE metrics such as the number of interruptions, bitrate quality switches, and average video quality, as an alternative to the network-related factors (discussed in Chapter 2.2). These objective QoE metrics values can be captured directly either with an intermediate measurement tool in the network or with a lightweight client-side instrumentation script. Subsequently, several objective metric-based parametric QoE models that aim at correctly determining a quality score making use of the aforementioned objective QoE metrics were proposed in the literature [17, 18, 19, 20]. Despite their promising features, the parametric QoE models in

the research literature have several limitations. We enumerate the limitations observed in the prior research as follows.

1. For end-point quality monitoring purposes, a parametric quality-estimation model should consider the clear insight of the user-related factors and individual preferences. Identifying the metrics that influence the QoE is vital to design models which are specially tailored to individual preferences. Such personalized models could improve the QoE by adapting bitrate (representation level) according to the user's preferences.

2. The QoE model must also consider the previous experience, which may be in the form of trained models, together with real-time application-level data collected from the client during the service usage to improve the accuracy of the quality score prediction.

3. The QoE model should have lower computational complexity to make it suitable for devices with constrained power and processing capabilities such as mobile phones.

4. Finally, the parametric QoE model that takes into account only application-level parameters (i.e., objective QoE metrics) will be reasonably simple to implement in mobile devices.

In addition, it is also crucial for the video service providers to attempt to resolve the central dilemma: maximizing resource utilization efficiency and revenues while maximizing end-users viewing experience. Moreover, offering an acceptable video QoE in mobiles stimulates the adoption of video services and enhances the end-user happiness and loyalty [162]. The service quality (i.e., QoS parameter) thresholds beyond which the QoE perceived by the end-users becomes unacceptable, the degree of influence and priority of each of the aforementioned objective QoE metrics on the end-user quality perception become indispensable differentiators for various stakeholders in the multimedia streaming process chain. The stakeholders must be aware of them in order to achieve a competitive advantage. To this end, there exists a serious need for the development of an accurate QoE-estimation model that tries to take into account the objective QoE metrics, a clear insight of the user-related factors and individual preferences to accurately predict the overall perceived QoE, which closely resembles the actual user-experienced video quality. This would also be helpful for the service providers to optimize their adaptation algorithm with an aim to

improve the overall viewing experience during a video session. In this chapter, we propose a parametric quality-estimation model based on multinomial logistic regression (MLR) to predict the overall QoE score for a HAS video session. Our proposed quality-estimation model is straightforward for the service provider to estimate/predict the QoE score of a video stream with ease.

4.2 Proposed Parametric QoE-Estimation Model

In this section, we discuss the multinomial logistic regression (MLR) model to predict the overall QoE for a DASH session. We present the different steps involved in the model, including consolidating the user's preferences, identifying the factors that affect the QoE, and training the model using the data annotated with the user's ratings. The block diagram in Fig. 4.1 depicts the steps involved in predicting the QoE score and the process flow involved. During Phase 1, we use the inputs from the subjective and objective experimental data to identify the factors that affect the QoE as perceived by the users. In Phase 2, we train the proposed parametric model with the objective QoE metrics collected at the client, annotated with user ratings to understand the user preferences. The trained model helps to determine the QoE score for a DASH session, given the objective metrics measured at the client. We briefly list the steps involved in the model before discussing them in detail.

Step 1: We first identify the subjective assessment method and prepare experimental design guidelines for performing a systematic subjective quality assessment (refer to Chapter 3.4.2).

Step 2: We conduct controlled subjective experiments where each of the test participants participated in several DASH video sessions (refer to Chapter 3.4.3). In each experiment, the test participant watches three videos and gives ratings that reflect their viewing experience. We obtain the subjective ratings on the ACR scale to determine the MOS for all the video sessions from these experiments.

Step 3: We also conduct an objective assessment using client-side instrumentation in the background for all the video sessions rated by the test participants during the subjective experiments. Apart from this, we also conduct the objective assessment for six more bitrate adaptation algorithms with three network trace profiles (discussed in Chapter 3.5.2). The objective assessment results with subjective scores (obtained with the three bitrate adaptation algorithms during the subjective QoE assessment) are subsequently used to train

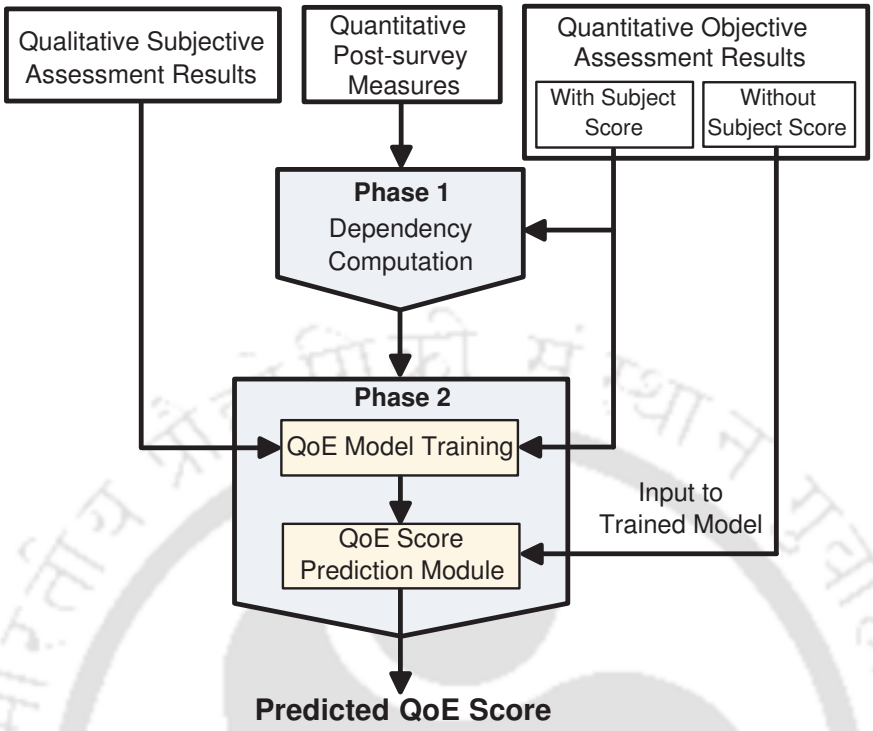


Figure 4.1: Sequence of steps in the proposed parametric QoE-estimation model.

the proposed model and validate the accuracy of QoE estimation.

Step 4: We also collect quantitative post-survey measures at the end of each subjective experiment that helps in understanding the user's preference/expectation with regards to the objective metrics during the video session (refer Chapter 3.6.1).

Step 5: To understand the influence of the objective QoE metrics on user's MOS ratings, we compute the dependency using both the quantitative objective assessment results and preferences from the quantitative post-survey stage. With this step, we only determine the objective QoE metrics that satisfy the global null hypothesis to train the proposed MLR model.

Step 6: The training module of the proposed quality-estimation model is based on the multinomial logistic regression. We use both subjective assessment results and the corresponding objective assessment results to train the proposed MLR model. The training module computes intercepts and regression coefficients for the independent variables (i.e., the five objective QoE metrics) using an iterative procedure. Then, it models the individual probabilities (for the response variables) using the multinomial log-odds (discussed in

Section 4.2.3).

Step 7: Finally, the objective assessment results (with a set of values corresponding to the objective QoE metrics) are given as an input to the trained model that predicts the QoE score for a video session.

4.2.1 Scope of Parametric Quality-Estimation Model

In recent years, several efforts have been made to understand the generic relationship between QoE and QoS in the research literature. However, these studies have mainly focused on a single/limited set of QoS parameters. Moreover, modeling the generic relationship between QoE and QoS parameters is not well understood and remained a challenging open problem. On the contrary, in the ever-increasing QoE parameter space, each of the QoE parameters adds a new dimension of complexity. Thus, systematically identifying the influencing QoE parameters and their relation to user-related factors is vital for designing a new QoE-estimation model that specifically targets individual end-users. According to the user profile, these personalized QoE models can further be utilized to improve the video QoE of an individual user by selecting appropriate video bitrate adaptation strategies. Accordingly, our parametric quality-estimation model first gets trained for a better understanding of the user-related factors and their influence on the set of QoE metrics. Subsequently, the set of video-related QoE metrics (as discussed in Chapter 3.5.1) are captured at the streaming client. The set of video-related QoE metrics are then provided as input to our trained QoE-estimation model. Thereupon, the QoE-estimation model outputs a QoE score that closely resembles the end-user viewing experience for the DASH streams.

4.2.2 Dependency Computation Procedure

The participants' preference/expectation for the corresponding objective QoE metrics (collected using the post-survey and presented in Chapter 3.6.1) and respective objective QoE metrics values were used to compute the dependency. It could be the case that a subset of the objective QoE metrics (i.e., independent variables) have only had their corresponding participants' preference/expectation feedback (i.e., dependent variables). The set of objective QoE metrics that only have participants' preference/expectation feedback can be tested whether the global null hypothesis is satisfied or not. On the other hand,

the rest of the objective QoE metrics (i.e., without participants' preferences) were directly considered for the model training phase.

We have considered a set of five objective QoE metrics, namely bitrate adaptation efficiency (BAE), bitrate switching amplitude reward (BSAR), interruption ratio (IR), initial startup delay reward (ISDR), and video continuity index (VCI) (as discussed in Chapter 3.5.1). In our case, we have only four metrics viz. BAE, BSAR, IR, and ISDR have the participants' preference/expectation feedback. The VCI metric without the participants' preference/expectation feedback is directly considered for the model training phase. At the same time, all the other four metrics, namely BAE, BSAR, IR, and ISDR are considered for the testing the global null hypothesis. Prior to that, we have computed the non-negative weighing parameter values as $\lambda = 3.87$, $\mu = 2.86$, $\rho = 3.38$, and $\sigma = 3.31$ correspond to the four objective QoE metrics viz. BAE, BSAR, IR, and ISDR, respectively (as discussed in Chapter 3.6.2). The larger the value of the non-negative weighing parameter, the higher is the influence of the objective QoE metric on the end-users QoE.

We describe the detailed procedure to determine the non-negative weight for each of the metrics mentioned earlier in Chapter 3.6.2. For completeness, we again present the equation used for computing the non-negative weights and is as follows.

$$Weight = \sum_{j} Numerical\_value(j) \times \left(\frac{P_j}{100}\right), \quad \forall j \in verbal\ phrase \tag{4.1}$$

where, $Numerical\_value(j)$ is the numerical value corresponds to the verbal phrase (as shown on Table 3.9) and P_j is the percentage of responses for respective verbal phrase (refer Table 3.10 and Table 3.11).

As a next step, these four non-negative weighing parameter values are rounded to whole numbers. The respective rounded parameters were then used as a reference or base category in order to calculate the dependency between an objective QoE metric and its influence on the viewer. We have carefully examined the global null hypothesis, i.e., likelihood ratio test, score test, and Wald test for all the four objective QoE metrics. We have observed that the p-value for each of the objective QoE metrics is less than 0.05 and the global null hypothesis is satisfied. Thus, we consider all five objective QoE metrics (including VCI) for model training.

4.2.3 Multinomial Logistic Regression-based Training Module

The subjective ratings provided by the test participants and their corresponding objective QoE metrics values (obtained during subjective quality assessment) are taken as input to train the proposed quality-estimation model. Our proposed quality-estimation model's outcome could take either of the five values from the subjective rating scale (i.e., 1 to 5) that closely resemble the end-users' viewing experience. Thus, we find the application of multinomial logistic regression (MLR) [163] is very relevant for the training module of our proposed parametric QoE-estimation model.

Multinomial logistic regression, also popularly known as *Softmax Regression*, is a supervised learning algorithm. Multinomial logistic regression is also known to be an efficient method and widely used for classification problems due to its state-of-the-art performance. The standard logistic regression model is mostly used in binary classification problems. The binary classification problems have response variable/outcome as dichotomous (e.g., distinguish between two kinds of hand-written digits) or binary in nature (e.g., yes/no). In contrast, multinomial logistic regression is a generalization of logistic regression for multi-class classification problems. In multi-class classification problems, the output/response can take more than two possible values. The multinomial logistic regression is suitable when the dependent variable (i.e., response variable) is categorical (nominal or ordinal), and the independent variables (i.e., predictive variables) are continuous or categorical. In general, the MLR model parameters are closely related to corresponding independent variables (i.e., predictive variables). If a predictive variable contributes a large amount to the classification, the corresponding model parameter value is also larger. Similarly, if a predictive variable contributes less to the classification, its corresponding model parameter value is also smaller.

Modeling the Logits: The MLR-based training module first computes intercepts and coefficients for all the independent variables (i.e., five objective QoE metrics). We nominate the subjective rating/score of five (i.e., one of the response categories) as baseline or reference cell. Next, we have computed the multinomial regression log-odds (logits) for all other categories relative to the baseline. Since the subjective rating scale has five response categories in total, our model has four log-odds. The log-odds of the four response categories are expressed as

$$log\left[\frac{P(1)}{P(5)}\right] = \beta_1 + \beta_{1,1}x_1 + \beta_{1,2}x_2 + \beta_{1,3}x_3 + \beta_{1,4}x_4 + \beta_{1,5}x_5. \quad (4.2)$$

$$log\left[\frac{P(2)}{P(5)}\right] = \beta_2 + \beta_{2,1}x_1 + \beta_{2,2}x_2 + \beta_{2,3}x_3 + \beta_{2,4}x_4 + \beta_{2,5}x_5. \quad (4.3)$$

$$log\left[\frac{P(3)}{P(5)}\right] = \beta_3 + \beta_{3,1}x_1 + \beta_{3,2}x_2 + \beta_{3,3}x_3 + \beta_{3,4}x_4 + \beta_{3,5}x_5. \quad (4.4)$$

$$log\left[\frac{P(4)}{P(5)}\right] = \beta_4 + \beta_{4,1}x_1 + \beta_{4,2}x_2 + \beta_{4,3}x_3 + \beta_{4,4}x_4 + \beta_{4,5}x_5. \quad (4.5)$$

In Eqs. (4.2) to (4.5), β_j is the intercept of the multinomial log-odds, for $j = 1, 2, ..., 4$. The intercepts for the response categories are estimated based on the training data. The x_1, x_2, x_3, x_4 and x_5 are the explanatory independent variables associated with BAE, IR, BSAR, VCI and ISDR, respectively. The $\beta_{1,1}$ to $\beta_{4,1}$, $\beta_{1,2}$ to $\beta_{4,2}$, $\beta_{1,3}$ to $\beta_{4,3}$, $\beta_{1,4}$ to $\beta_{4,4}$, and $\beta_{1,5}$ to $\beta_{4,5}$ in Eqs. (4.2)-(4.5) are the regression coefficients of the four log-odds for BAE, IR, BSAR, VCI and ISDR, respectively. The regression coefficients in Eqs. (4.2)-(4.5) are represented as $\beta_{j,k}$ and are associated with the jth probability outcome and kth independent variable. The multinomial logistic regression uses an iterative procedure, called maximum likelihood estimation algorithm, to compute the regression coefficients. The proposed quality-estimation model computes the regression coefficients using the training data samples (containing both subjective rating and corresponding five objective QoE metric values). In the first iteration (called iteration 0), the model has no regression coefficients and is called as log likelihood of the "null" model. In stage 1, the initial values are set for all the coefficients of the five metrics, i.e., β_{11} to β_{41}, β_{12} to β_{42}, β_{13} to β_{43}, β_{14} to β_{44}, and β_{15} to β_{45}. In the next iteration, the regression coefficient(s) are included and optimized using the training data samples. At each iteration, the log likelihood decreases because the goal is to minimize the log likelihood. The process is repeated until the difference of log likelihood between successive iterations is very small. Once the residual deviance is minimized, the model regression coefficient(s) is/are said to have "converged". We assume the right-hand side (R.H.S.) of the log-odds in Eqs. (4.2) to (4.5) are denoted as Y1, Y2, Y3, and Y4, respectively. Next, we recover the odds in Eqs. (4.2) through (4.5) by exponentiating both

sides. They can be written in the following mathematical form.

$$\frac{P(1)}{P(5)} = e^{Y1}. \tag{4.6}$$

$$\frac{P(2)}{P(5)} = e^{Y2}. \tag{4.7}$$

$$\frac{P(3)}{P(5)} = e^{Y3}. \tag{4.8}$$

$$\frac{P(4)}{P(5)} = e^{Y4}. \tag{4.9}$$

Modeling the Probabilities: The individual probabilities are computed using the above multinomial log-odds, i.e., Eqs. (4.6) to (4.9). By combining the Eqs. (4.6) to (4.9), we get

$$\frac{P(1)+P(2)+P(3)+P(4)}{P(5)} = e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}. \tag{4.10}$$

According to probability theory, we know that

$$P(1)+P(2)+P(3)+P(4)+P(5) = 1, \tag{4.11}$$

We can re-write the Eq. (4.10) to obtain the individual probability for the baseline as

$$\frac{1-P(5)}{P(5)} = e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}. \tag{4.12}$$

$$\frac{1}{P(5)} = 1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}. \tag{4.13}$$

$$P(5) = \frac{1}{1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}}. \tag{4.14}$$

Next, we substitute the Eq. (4.14) in Eqs. (4.6) through (4.9) in order to get the individual probabilities for the other response categories. The close mathematical form for individual probabilities of the other response categories are as follows.

$$P(1) = \frac{e^{Y1}}{1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}}. \tag{4.15}$$

$$P(2) = \frac{e^{Y2}}{1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}}. \tag{4.16}$$

$$P(3) = \frac{e^{Y3}}{1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}}. \tag{4.17}$$

$$P(4) = \frac{e^{Y4}}{1 + e^{Y1} + e^{Y2} + e^{Y3} + e^{Y4}}. \tag{4.18}$$

QoE Score Prediction: The trained MLR model predicts the QoE score for a given input comprising of the objective QoE metrics values. The trained model first computes the individual probabilities for the five response categories, using the Eqs. (4.14) through (4.18), using the input provided. Next, the response category with the highest probability is predicted to be the QoE score for the given input.

4.3 Performance Evaluation

In this section, we first present the implementation details of the proposed model for QoE prediction. Next, we briefly discuss the procedure followed to train the proposed MLR model using the training dataset. We then perform the quality-estimation accuracy validation of our proposed MLR model using the validation dataset. We also evaluate the efficiency of the proposed MLR model by comparing it with other baseline QoE models. For additional context, we also predict the QoE score for the DASH bitrate adaptation algorithms merely providing the set of objective QoE metrics values collected from the extensive experimentation (refer Chapter 3.5.2) as input to the proposed MLR model. Finally, we also demonstrate the applicability of our proposed quality-estimation model for longer video sequences.

4.3.1 Implementation Details

The QoE model should always provide insights to various stakeholders, such as network providers and video service providers, about the end-users' viewing experience. In turn, these insights can be helpful to the stakeholders for optimized system design and efficient QoE management. Accordingly, we have implemented our proposed multinomial logistic regression-based QoE-estimation model in R. The proposed MLR-based QoE-estimation model can be deployed either on a PC or a tablet. Note that we have used a PC configured with Windows 7 OS for both training and prediction accuracy validation.

4.3.2 Procedure for Parametric MLR Model Training and Testing

For model training and testing, we use both hold-out and k-fold cross-validation methods. In the hold-out method, we used 80% of the data from the subjective quality assessment (discussed in Chapter 3.4.3) for training and the rest for testing. Thus, the dataset comprising a total of 261 tuples was split into two sets. The training set with 209 entries is used for the training purpose and the test set with 52 entries is used for validation of prediction accuracy. We computed the regression coefficients for our proposed MLR model using the training set with 209 entries (obtained using the hold-out method). The observed values for the regression coefficients are reported in Table 4.1.

Table 4.1: Regression coefficient values obtained with the training data.

Metrics	Coefficients	Value
BAE	$\beta_{1,1}$	0.193541
	$\beta_{2,1}$	0.216484
	$\beta_{3,1}$	0.210127
	$\beta_{4,1}$	0.709242
IR	$\beta_{1,2}$	1.13244
	$\beta_{2,2}$	1.7002401
	$\beta_{3,2}$	0.91754
	$\beta_{4,2}$	1.0127475
BSAR	$\beta_{1,3}$	3.15470
	$\beta_{2,3}$	1.24578
	$\beta_{3,3}$	2.95468
	$\beta_{4,3}$	0.854801
VCI	$\beta_{1,4}$	0.684875
	$\beta_{2,4}$	1.445770
	$\beta_{2,4}$	1.9784734
	$\beta_{2,4}$	2.1425701
ISDR	$\beta_{1,5}$	0.00782620
	$\beta_{2,5}$	-0.03374961
	$\beta_{3,5}$	-0.04661514
	$\beta_{4,5}$	0.00301220

Next, we used k-fold cross-validation with five independent splits of 80%-20% as training-test folds with grouping at the individual test participant level. In other words, for any given split, no data from the same test participant are in both the training and testing folds. Accordingly, each split will have a total of 51 data points that belong to seventeen test participants. The test participants for each split are chosen randomly from

the pool of eighty-seven participants. Note that the selection of the number of splits in k-fold cross-validation is based on performance on the collected dataset and to ensure the 80%-20% partition for training-test folds. The general procedure for the k-fold cross-validation is to use K-1 splits (out of the K splits) for training purpose, whereas the remaining split is used for testing purpose. The proposed MLR model is then trained and tested K times; each time, a new split was used as a testing set while the remaining splits were used for training. We report the results obtained for the proposed MLR-based QoE-estimation model with multiple train-test splits in Section 4.3.4.

4.3.3 Validation of Prediction Accuracy

The key performance evaluation criteria of a QoE-estimation model include prediction accuracy, as specified by the in Video Quality Experts Group [164]. The prediction accuracy measures how accurately (or with low error) the trained model is able to predict the subjective scores using the objective QoE metrics values. The proposed MLR model's efficiency was demonstrated using the root mean square error (RMSE) and Pearson's linear correlation coefficient (PLCC) as a performance index. The RMSE is formally defined in Eq. (4.19).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y - \hat{y})^2}{n}}, \tag{4.19}$$

where y is the output of the actual experience rating, $\hat{y}$ is the estimated model output and n is the length of the validation dataset.

Unlike the P.1201 model [165], the proposed MLR-based QoE-estimation model does not rely on a video-frame-type estimation module. However, we have used the RMSE and PLCC values of the P.1201 model, i.e., "RMSE $\leq$ 0.65" and "PLCC $\geq$ 0.70", as the minimum performance benchmark to verify the quality prediction accuracy of our proposed MLR-based QoE-estimation model. Moreover, the degree of prediction accuracy for the QoE-estimation model will definitely affect its applicability during the QoE management process.

It is evident from Table 4.2 that the proposed MLR-based QoE-estimation model was well trained. We have observed that PLCC and RMSE values were nearly the same for both training and test sets. As reported in Table 4.2, the RMSE and PLCC values observed for

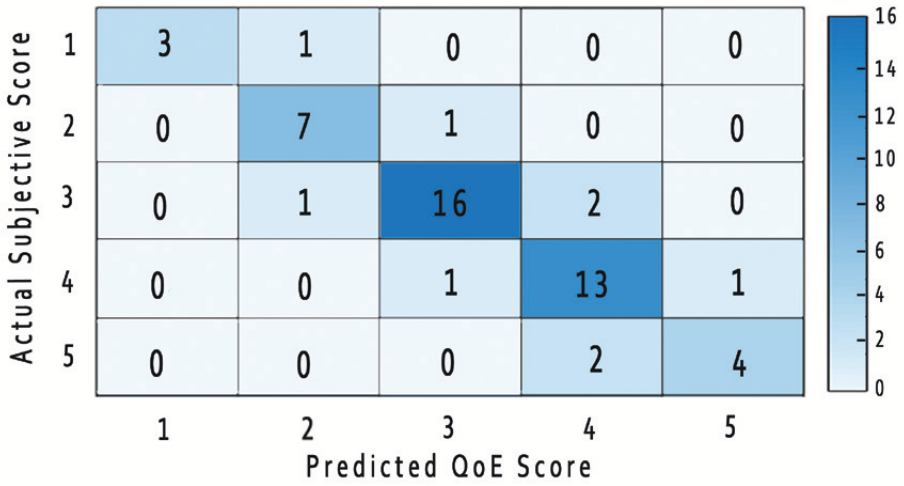


Figure 4.2: Confusion matrix for the samples of test dataset.

the test set outperforms the minimum performance benchmarks. From Table 4.2, we can observe that the proposed model achieves a sufficiently high PLCC value of 0.9235 in the model validation process. Thus, we conclude that our proposed MLR-based QoE-estimation model can accurately predict the overall viewing experience for DASH streams in mobile clients.

Table 4.2: The RMSE and PLCC calculated for training data and validation data.

	Training	**Validation**
RMSE	0.4913	0.4160
PLCC	0.8316	0.9235

Fig. 4.2 illustrates the prediction confusion matrix for the results of the validation phase. For every cell in the confusion matrix, the row represents the actual subjective rating/score and the column represents the model predicted QoE score. The diagonal cells in the matrix represent the ideal case where the instance is predicted accurately. The miss predicted instances by the proposed MLR-based QoE-estimation model are represented by all the off-diagonal cells in Fig. 4.2. We have computed the proposed QoE-estimation model accuracy as 83% using the confusion matrix (as shown in Fig. 4.2). The relationship between the actual values of the subjective rating and the model predicted QoE scores for all the video session samples in the test set is shown in Fig. 4.3.

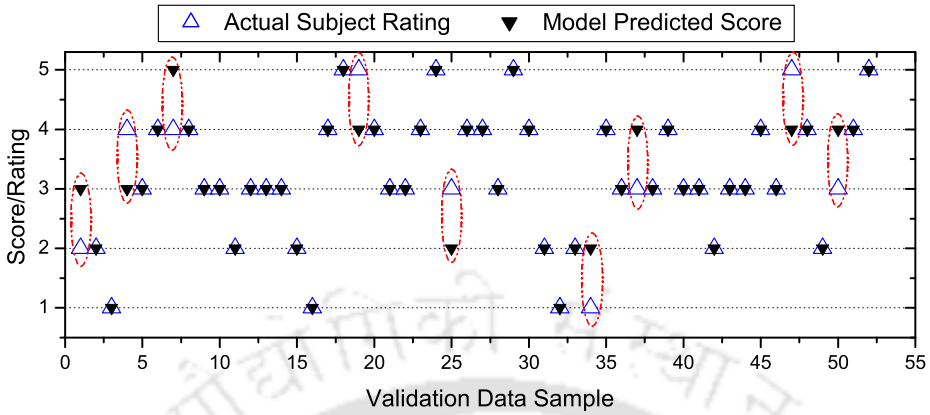
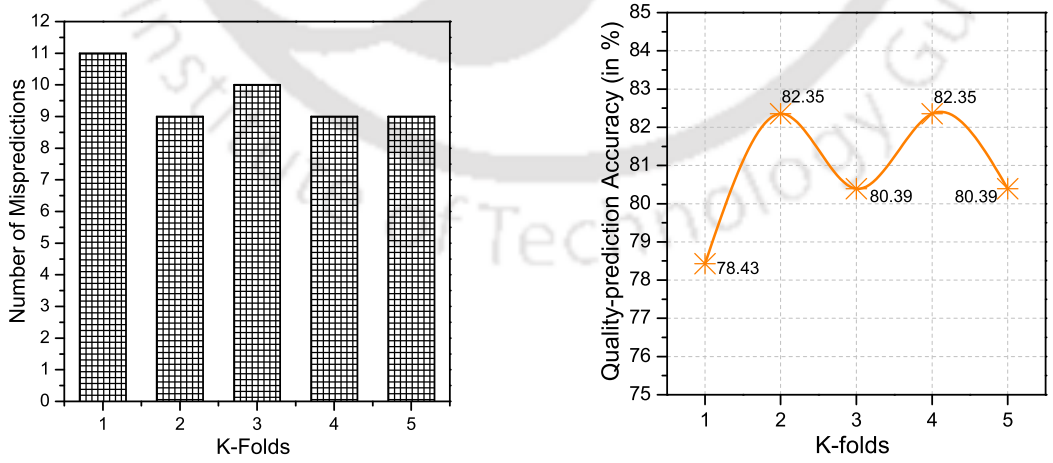


Figure 4.3: Model predicted quality score vs. actual subject rating for the samples of test dataset.

4.3.4 Analysis of Five-Fold Cross-Validation

Next, we use a five-fold cross-validation technique to validate the model and the accuracies with various k values are reported in Table 4.3. The number of mispredictions observed for the different test folds (with 51 samples) is shown in Fig. 4.4a. The highest prediction accuracy was 82.35% and observed at both fold 2 and fold 4 as illustrated in Fig. 4.4b. The proposed MLR-based QoE-estimation model achieved a mean prediction accuracy of



(a) (b)

Figure 4.4: Observations from five-fold cross-validation: (a) number of mispredictions and (b) prediction accuracies achieved with different folds.

80.78% with a standard deviation of 1.641 based on the 5-fold cross-validation (as reported in Table 4.3).

Table 4.3: Quality prediction accuracy observed with five-fold cross-validation.

K-Fold	Prediction Accuracy (in percentage)	Mean Prediction Accuracy (with Standard Deviation)
1	78.431	80.784 (1.641)
2	82.351	
3	80.392	
4	82.352	
5	80.392	

4.3.5 Impact of Training Dataset Size

In this subsection, we used the training and testing folds obtained with the hold-out method to assess the impact of the training fold size. The 20% testing fold with 52 samples is used for quality-estimation accuracy validation. On the other hand, we have prepared five different training datasets with 90, 120, 150, 180, and 209 samples, respectively, for the model training purpose in order to assess the impact of training dataset size on prediction accuracy. We computed the regression coefficient sets for each of the five different training datasets. Subsequently, our proposed MLR-based QoE-estimation model was separately trained with the five different training datasets. The trained models were then used to predict the quality score for the 52 samples in the test set. The analysis of the model predicted quality scores and actual subjective ratings for the five different training datasets is shown in Fig. 4.5. The observed values demonstrate that the number of mispredictions is reduced with the increase in the training sample size (as shown in Fig. 4.5a. The steady increase in the quality-prediction accuracy, shown in Fig. 4.5b, is attributed to the decrease in the number of mispredictions with the increase in training sample size. The RMSE and PLCC results obtained with different training dataset sizes (as shown in Fig. 4.5c and Fig. 4.5d, respectively) demonstrate that our proposed model performs better in predicting the quality score with the increase in the training fold size. It is also evident from Fig. 4.5d that our proposed model has achieved a high PLCC value of 0.83056 as compared to the minimum performance benchmark (i.e., "PLCC $\geq$ 0.70") even when trained with a training set consisting of 90 samples.

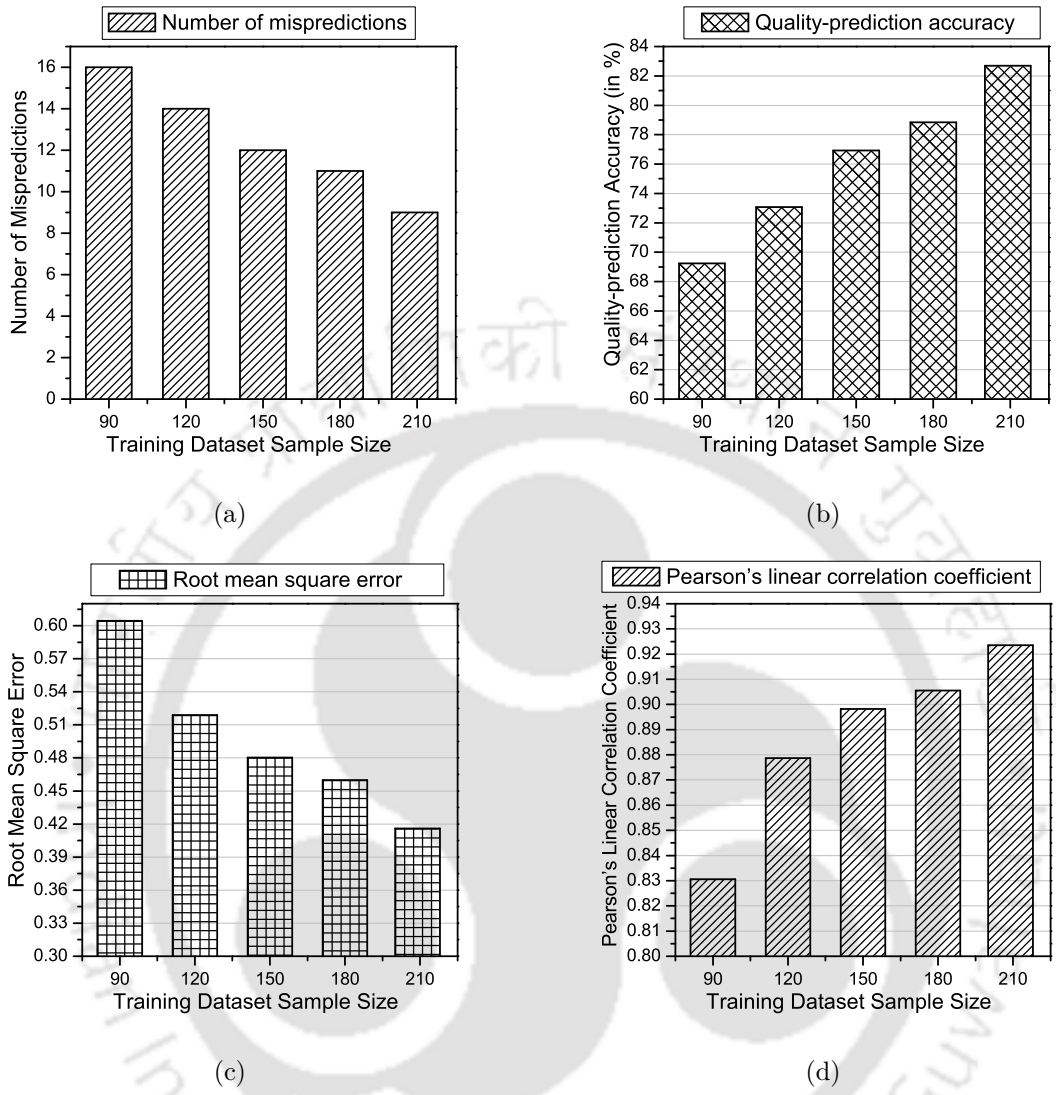


Figure 4.5: Observed values with different training dataset sample sizes: (a) number of mispredictions, (b) quality-prediction accuracy, (c) RMSE and (d) PLCC.

4.3.6 Model Comparison

We compare our proposed MLR-based QoE-estimation model results with three existing baseline QoE models in terms of the prediction accuracy and computation complexity. The three baseline models considered for comparison include the linear parametric QoE scoring model (as discussed in Chapter 3.6) and two variants of the general QoE metric used by [36]. The linear parametric QoE scoring model utilizes the objective QoE metrics and the non-negative weights assigned to the respective objective QoE metrics to compute a cumulative QoE score. For a better readability, we again formally defined the linear

parametric QoE scoring model (described in Chapter 3.6) in Eq. (4.20).

$$QoE_{param} = (3.87 \times BAE) + (2.86 \times BSAR) + (3.38 \times (1 - IR)) + (3.31 \times ISDR) + VCI. \tag{4.20}$$

The general QoE metric used by [36] is defined as:

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)|, \tag{4.21}$$

for a video with N segments. R_n represents the bitrate of nth segment and $q(R_n)$ maps the R_n to the quality perceived by the end-user. T_n and μ represent the rebuffering time resulted from downloading nth segment and rebuffer penalty, respectively. The last term in Eq. (4.21) represents the penalty for change in video bitrate for two adjacent segments in order to favor smoothness during the video session.

For model comparison, we consider two choices of $q(R_n)$:

1. QoE_{lin}: Here, $q(R_n) = R_n$ and the rebuffer penalty (μ) is set to 4.3. This QoE metric is used by [68].

2. QoE_{log}: Here, $q(R_n) = log(R/R_{min})$ and the rebuffer penalty (μ) is set as 2.66. This QoE metric is used by [37] and captures the marginal improvement in perceived video quality by the viewer while streaming at higher bitrates.

In order to make a fair performance comparison of the models, we have normalized the QoE scores obtained using the aforesaid three QoE models (i.e., QoE_{param}, QoE_{lin} and QoE_{log}) onto a 1 to 5 scale. For the evaluation, we have considered the 5-fold cross-validation dataset (discussed in Section 4.3.2). We used the four folds for training our proposed MLR-based QoE-estimation model. Subsequently, the remaining test fold (with 51 samples) was used to predict/compute the quality score by all four models. This procedure is repeated five times with different test folds. The average of the observed results were used for the model comparison. We have used the Pearson's linear correlation coefficient (PLCC) and root mean squared error (RMSE) as the performance metrics to compare the proposed MLR-based QoE-estimation model with the other three models. We have also measured the computation complexity of all the models as the time required to predict/compute the quality score with a raw input trace (excluding the time required for the training phase,

if any). It should be noted that the measurement of the computation complexity was conducted on a PC with an Intel Core i7 processor running at 3.40GHz and 4GB RAM.

The performances and the computation complexity of the four models are summarized in Table 4.4. In Table 4.4, the mean of the PLCC and RMSE observed over the five test folds (each containing 51 samples) and the mean of the computation complexity measured for all the samples in five test folds were reported. It is evident from Table 4.4 that our proposed MLR-based QoE-estimation model outperforms all the other models in terms of PLCC and RMSE value. In general, PLCC (close to 1) indicates a superior correlation with the actual participants' experience ratings. A lower value observed for RMSE also indicates a higher performance by our MLR-based QoE-estimation model. The computation complexity of our proposed model is considerably higher as compared to the others. It can be observed that the MLR-based QoE-estimation model only takes less than 0.1 sec to predict the quality score. Meanwhile, the other three models used for comparison have an average processing time around in the range of 0.04 sec to 0.06 sec per quality computation. Hence, the proposed MLR-based QoE-estimation model can be appropriately applied to real-time video quality monitoring.

Table 4.4: Models performance in predicting quality score.

Model	**Performance Index**		**Computation Complexity (in ms)**
	PLCC	**RMSE**	
QoE_{param}	0.82634	0.74328	65.629
QoE_{lin}	0.90447	0.55228	48.864
QoE_{log}	0.90686	0.54681	56.273
MLR model	0.93151	0.46345	71.956

The scatter plots of quality scores predicted by the MLR-based QoE-estimation model, QoE_{param} model, QoE_{lin} model and QoE_{log} model against the actual participant ratings for the test fold 1 are shown in Fig 4.6. It is clearly evident from Fig 4.6a that our proposed model exhibits a better convergence and monotonicity performance as compared to the other models. Thus, we can conclude that our proposed MLR-based QoE-estimation model demonstrates a superior prediction accuracy than the other models.

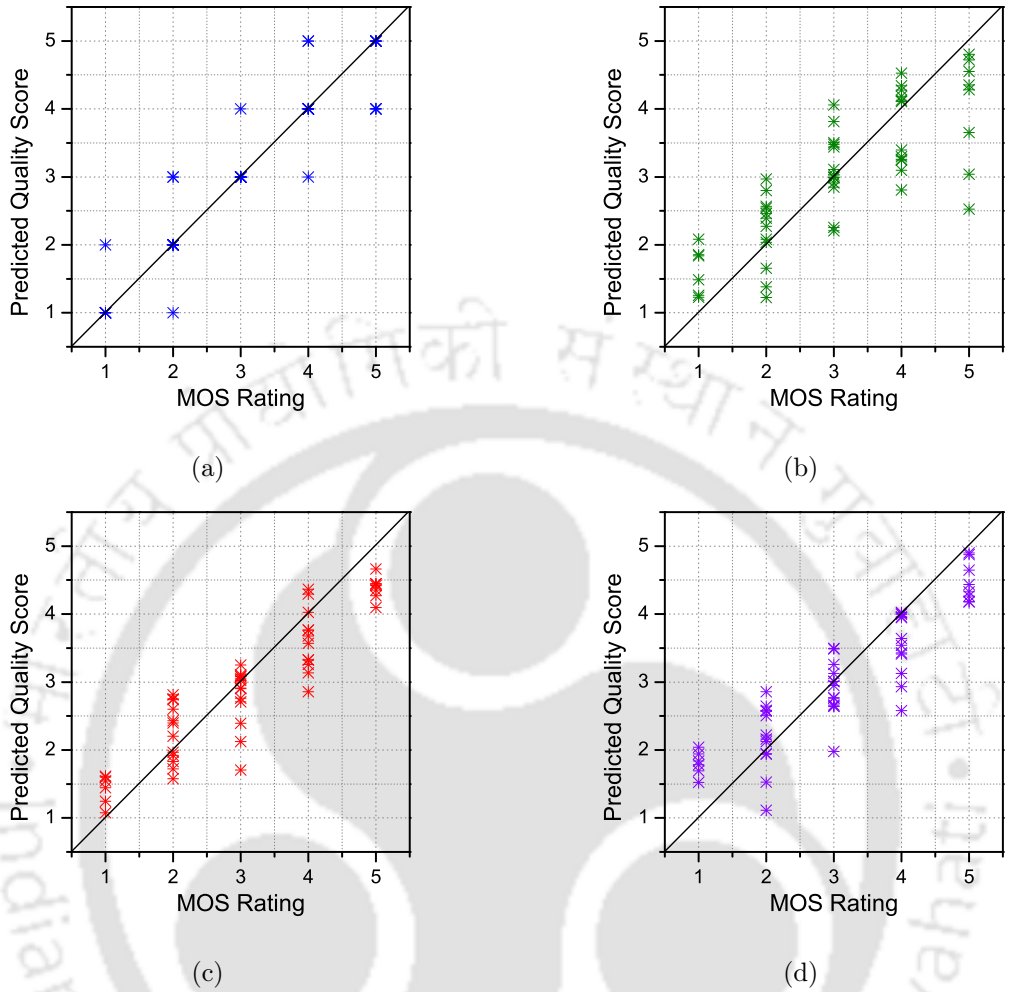


Figure 4.6: Actual MOS rating vs. model predicted quality score: (a) proposed MLR model (b) Parametric QoE model, (c) QoE linear metric and (d) QoE log metric.

4.3.7 Model Predicted QoE Score for DASH Adaptation Algorithms

We present the predicted QoE score estimated by the proposed MLR-based QoE-estimation model for the six different bitrate adaptation algorithms, where we had only objective assessment data, merely using the measured objective QoE metrics as input. The individual QoE score estimated using the objective QoE metrics input instances were averaged to calculate the overall user experience for each of the test scenarios. The mean of estimated QoE scores for all the algorithm-trace combinations with the three video sequences are reported in Table 4.5. From Table 4.5, we observe that the DASH adaptation algorithms that consider both playback buffer occupancy and throughput estimation for segment bitrate decision making outperforms all others in terms of the predicted QoE score. The higher

predicted QoE score for these adaptation algorithms can be attributed to the higher values recorded by the objective QoE metrics (deduced by comparing the objective assessment results in Table 3.6, Table 3.7 and Table 3.8 as reported in Chapter 3.5.2).

Table 4.5: The MLR-based model predicted QoE score for the bitrate adaptation algorithms.

Adaptation Algorithms	Estimated QoE Score*								
	Big Buck Bunny			Red Bull Play Street			Tears of Steel		
	Train	Car	Ferry	Train	Car	Ferry	Train	Car	Ferry
BOLA [37]	3.5833	3.5833	3.4167	3.3333	3.4167	3.0833	3.5833	3.4167	3.3333
ABMA+ [40]	3.0833	2.8333	3.3333	3.1667	3.1667	2.9167	3.5833	3.1667	3.6667
QAAD [87]	2.6667	2.4167	2.2500	2.5833	2.4167	1.9167	2.9167	2.6667	2.4167
ARA [80]	2.3333	2.2500	2.1667	2.1667	2.0833	2.0833	2.4167	2.9167	2.1667
KM [79]	3.1667	3.2500	3.3333	3.4167	3.3333	2.9167	3.6667	3.3333	3.0833
FSRA [74]	1.8333	2.1667	1.9167	1.8333	2.0833	2.1667	2.5833	2.4167	2.5833

*The average QoE score is measured or estimated out of the multiple measurements made in the different iterations under specific network conditions (scenario). We report the estimated average QoE score for all the trace-algorithm combinations.

4.3.8 Model Applicability for Longer Duration DASH Videos

One of the limitations of our proposed MLR-based QoE-estimation model is that the model was trained using the data obtained from 8-minute long video sequences. We choose the limit on the duration of the test video sequence to limit the duration of the video session that avoids user fatigue in subjective experiments. Furthermore, according to the ComScore online video rankings survey, the average duration of online video content is about 6.4 minutes long [166]. Therefore, the proposed MLR model can be directly applied to predict the quality score for most of the online videos. Moreover, our proposed model only takes the objective QoE metrics values as input and these values can be effortlessly recorded using lightweight client-side instrumentation. Accordingly, we propose two different approaches to estimate the overall end-user viewing experience for DASH videos of long duration. First, we propose to compute the objective QoE metrics for the entire video session. Subsequently, estimate the QoE score to capture the user experience at the end of the video session. However, this can significantly fail to capture the end-users' viewing experience at different stages of the video session and the impact of the bandwidth fluctuations. Alternatively, we propose to split the entire video into either eight-minutes or smaller intervals. Then record the objective QoE metrics values for each of the intervals. The concepts of continuous quality mentioned in ITU Recommendation BT.500-13 [158] and widely investigated in

the recent literature, especially in the context of Web services and video streaming. To demonstrate the application of the proposed MLR model for longer videos, we use three different concepts of quality score. The definitions of the different concepts of scoring are as follows.

1. *Continuous quality score*: This is defined as the instantaneous quality score estimated at the end of the video interval using the objective QoE metrics captured during the interval.

2. *Cumulative quality score*: This is defined as a quality score estimated making use of the cumulated objective QoE metrics recorded from the beginning of the video session up to the end of any interval of the session.

3. *Overall quality score*: This is defined as the mean of the cumulative quality score estimated at the end of each interval up to the current interval.

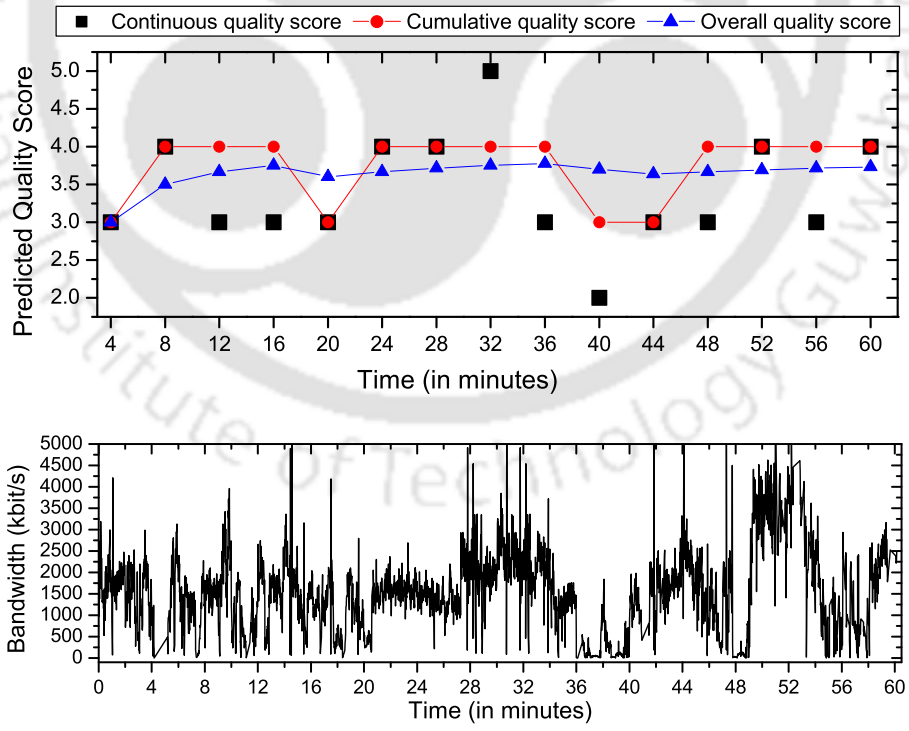


Figure 4.7: Temporal variation of the estimated QoE scores for 60-minute Red Bull PlayStreets video with the corresponding network bandwidth variation.

Accordingly, we have conducted the experiment with a 60-minute long video. We consider the Red Bull PlayStreets video sequence restricted to a 60-minute play duration and the SARA adaptation algorithm [39] for conducting the experiments. The underlying network link bandwidth is emulated with both train and ferry trace (each comprises of 30-minute long bandwidth estimations) during the experiments. Instead of providing a single QoE score for the entire video, we considered an interval of 4-minutes and used the MLR-based QoE-estimation model to predict three types of quality scores defined earlier. Fig. 4.7 depicts the three different quality scores estimated using our MLR-based QoE-estimation model. We observe from Fig. 4.7 that both the continuous quality score and cumulative quality score have a high correlation with the underlying network bandwidth profile (considered for the link emulation) and shown right below the plot. From Fig. 4.7, it is evident that all the proposed concepts of quality score estimation have approximately resulted in a similar quality score for the 60-minute long video sequence. Thus, instead of just predicting the single quality score, our model could also be used to predict the score in real-time for the entire video session. The predicted quality score at every individual interval could also provide a clear insight into the QoE changes perceived by the viewer. In additional context, the bitrate adaptation algorithm employed at the client could leverage this information to make an informed segment bitrate decision(s) to improve the overall viewing experience.

4.4 Summary

In this chapter, we proposed a parametric multinomial logistic regression-based QoE model to estimate the overall user experience for DASH streams in mobile clients. We have considered the individual subject ratings and five objective QoE metrics values captured during the subjective and objective quality assessment, respectively, as discussed in Chapter 3. We compute the dependency using both the quantitative objective results captured during subjective experiments and preferences from the quantitative post-survey stage in order to understand the influence of the objective QoE metrics on the end user's viewing experience. We then trained the proposed MLR-based QoE-estimation model with the subjective assessment results and the corresponding objective assessment results. The trained MLR-based QoE-estimation model only requires access to a set of objective QoE metrics values,

captured with a lightweight script that can be incorporated into the DASH media player even in mobile clients, as input for the QoE score prediction. We have also validated our proposed model's quality-estimation accuracy using both hold-out and k-fold cross-validation training-test sets. The quality-estimation accuracy validation results underline the proposed model's high performance in achieving a PLCC of 0.9235 and an RSME of 0.416. Our proposed model also achieved 83 percent prediction accuracy with the hold-out test set. The experimental results from the comparison with three other benchmarking models in the literature have demonstrated that the proposed model shows better prediction accuracy with a reasonably low computational complexity, suggesting its suitability for online use. In the next chapter, we present an adaptive segment-aware K-Push algorithm to determine the appropriate push adaptation pair in DASH/2 while mitigating the request explosion problem and addressing the over-push problem in mobile clients.

CHAPTER

5

Towards Improved QoE with Adaptive Segment-aware K-Push Algorithm for Dynamic Adaptive Streaming over HTTP/2

The previous chapters have focused on assessing and modeling the quality of experience for Dynamic Adaptive Streaming over HTTP (DASH), which is the dominant streaming technology in Video-On-Demand (VoD) scenarios in mobile devices operating under dynamic network conditions. Despite its good performance and being the most widely deployed technology, DASH suffers from the request message overhead (also termed as *request explosion* problem) in mobile networks. This problem gets aggravated as the request messages, especially in scenarios with multiple streaming clients, compete for the constrained uplink bandwidth (as discussed in Chapter 2.3). This could significantly delay the video segment download, resulting in frequent playback interruptions and selecting lower video bitrate for future segments. Thus, the viewer could have a significantly degraded QoE. To mitigate the *request explosion* problem and improve the QoE for the DASH session, this chapter presents an adaptive segment-aware K-push algorithm leveraging the *Server Push* feature of HTTP/2 for DASH segment delivery. The proposed K-push algorithm decides the suitable adaptation push pair (i.e., number of segments and their corresponding video bitrate level) on runtime. The adaptation push pair decision is based on estimated network throughput, current playback buffer occupancy, and segment size information. In this chapter, the proposed adaptive segment-aware K-push algorithm's design is detailed and compared to other existing adaptation algorithms under various mobile video streaming scenarios. Our extensive experimental results demonstrate that the proposed approach significantly

increases the bitrate adaptation efficiency and reduces interruptions compared to existing benchmarking algorithms.

Organization of the Chapter: The remainder of this chapter is structured as follows. In Section 5.1, we present the motivation for the work. Next, we present the full description of the proposed adaptive segment-aware K-push algorithm in Section 5.2. We then summarize comprehensive implementation details of our proposed DASH/2 prototype and trace-driven network emulation setup in Section 5.3. In Section 5.4, we evaluate our proposed algorithm and discuss its benefits when compared to existing heuristics in the literature. The chapter is concluded in Section 5.5.

5.1 Motivation

In recent years, leveraging the HTTP/2 server-initiated push mechanism for Internet streaming delivery has attracted several research efforts. A few researchers designed K-push schemes for both low latency live and DASH-compliant video streaming [23, 24, 25, 26, 27]. Mueller et al. in [99] implemented MPEG-DASH over HTTP/2 (i.e., SPDY) and demonstrated its potential benefits and drawbacks. Furthermore, they have also carried out experimental evaluations to investigate link bandwidth utilization with different RTT values and found a noticeable protocol overhead incurred due to Secure Socket Layer (SSL) encryption. The authors in [167] have investigated the server push mechanism for a fast start in the DASH-compliant video streaming. Recently, the work in [26] have investigated the potential of *fixed K-push scheme* for DASH segment delivery over HTTP/2 (DASH/2). In every push cycle, the streaming client determines the video bitrate level $\hat{b}_\kappa$ and push length of fixed κ segments. The client sends a request together with the *Push Directive* (i.e., $\hat{b}_\kappa$ and κ) to the streaming server. Subsequently, the steaming server initiates a new push cycle session. Thereafter, the client receives κ segments that are pushed by the HTTP/2 server without receiving the corresponding segment requests. They have observed significant improvement in terms of underlying network bandwidth utilization compared to legacy HTTP/1.1. However, their analysis of *fixed K-push scheme* also uncovers several key challenges: 1) causes the over-push problem; 2) leads to degraded network adaptability; and 3) no perceptible improvement in streaming throughput (i.e., segment quality variation) as

κ value is increased. The *request explosion* problem could be greatly reduced by determining a large value for push number (the number of segments to be pushed) in a push cycle of DASH/2. However, if the viewer quits watching the video at an early stage, resources (from both server and network) used for content push gets wasted [8]. It is also evident that both the number of segments (κ) and their corresponding segment bitrate level in a push cycle have mutual dependence. Selecting fewer segments for a push cycle could lead to increased *request explosion*. On the contrary, a lower bitrate level for the video segments in a push cycle could result in notable degradation in the end user's quality of experience. Hence, delivery of video segments using DASH/2 is not straightforward and it is vital to determine both the push number and respective segment bitrate level appropriately.

Few works in the literature, to address the limitations mentioned above, have proposed to dynamically scale the push number and determine the appropriate adaptation push pair (κ, $\hat{b}_\kappa$) - κ is the number of segments to be pushed in the current push cycle and $\hat{b}_\kappa$ is the video bitrate level selected - at runtime [26, 27]. Through real-world trace-driven experiments, they have demonstrated that adaptive-push significantly improved both network utilization and adaptability while alleviating the over-push problem.

From a streaming point of view, we provide some critical observations on DASH that poses challenges in DASH/2 performance optimization. The segments of a DASH video have the same playback duration (typically ranging from 1 - 15 seconds). However, it was observed that their sizes across the same bitrate level could vary significantly. The maximum, minimum and average segment size for various video bitrate levels (or representations) of Big Buck Bunny, Tears of Steel, and Red Bull Playstreets video sequences (with 10 second segment play duration) from DASH video dataset [153] are reported in Table 5.1. For example, in the Big Buck Bunny video sequence with 3.5 Mbps bitrate, we observe that the segment size varies from 1.3 MB to 7.2 MB, with the average segment size being 4.1 MB. A similar variation in segment size is also seen for the segments of other video sequences reported in Table 5.1.

In DASH, bitrate adaptation mechanisms usually take the segment download rate estimation into account, aiming to match the estimated available bandwidth to the playout bitrate. Thus, the estimation of the available bandwidth for the next segment(s) is crucial to the user-perceived QoE. In general, the DASH clients make use of the segment timestamps

Table 5.1: Maximum, minimum and average segment sizes (in KBytes) for the respective video bitrate levels of Big Buck Bunny, Tears of Steel, and Red Bull Playstreets.

Video Sequence Name	Video Bitrate	Segment Size (in KBytes)		
		Max	Min	Mean (SD)
Big Buck Bunny	3.5 Mbps	7200	1300	4145 (1495)
	2.4 Mbps	3600	1300	2820 (566)
	1.4 Mbps	2400	664	1700 (441)
	0.6 Mbps	1100	649	930 (92)
Tears of Steel	6.0 Mbps	15000	2200	7377 (2449)
	4.0 Mbps	9900	1100	4926 (1671)
	2.9 Mbps	7300	727	3694 (1295)
	0.8 Mbps	1900	223	990 (343)
Red Bull Playstreets	5.9 Mbps	7900	3900	6780 (790)
	4.0 Mbps	5100	2700	4731 (290)
	2.4 Mbps	3300	1600	2806 (361)
	1.4 Mbps	1900	1000	1778 (109)

to estimate the segment download rate. These estimations could vary significantly as they are closely dependent on client hardware configuration and streaming environment. To get a better understanding of the download rate estimation, we show the impact of segment size variation in DASH user space with a mobile device. To this end, we have used a streaming testbed setup with a client-server topology, where the server and mobile client are connected to a wireless access point (AP). We have throttled the bandwidth of the link between AP and mobile client with a link capacity of 5 Mbps and we do not have any background traffic as well. To capture the download rate during the segment download process, we have done custom modifications in the client media player (i.e., `dash.js`) code. The Big Buck Bunny video sequence with bitrate set to 3.5 Mbps and segment duration of 10 seconds is streamed. Fig. 5.1 depicts the average of the estimated download rate (obtained from multiple runs) for video segments of different sizes with 95% confidence intervals. From Fig. 5.1, we deduce that the DASH download rate estimate is indeed affected by the variation in segment size. We can also observe that the inaccuracy of the download rate estimate minimizes with the increase in the segment size (as shown in Fig. 5.1).

In addition, the variation in the segment download times (for various video bitrate levels) of three different video sequences, namely Big Buck Bunny, Tears of Steel, and Red Bull Playstreets with 10 second segment play duration from the DASH video dataset [153]

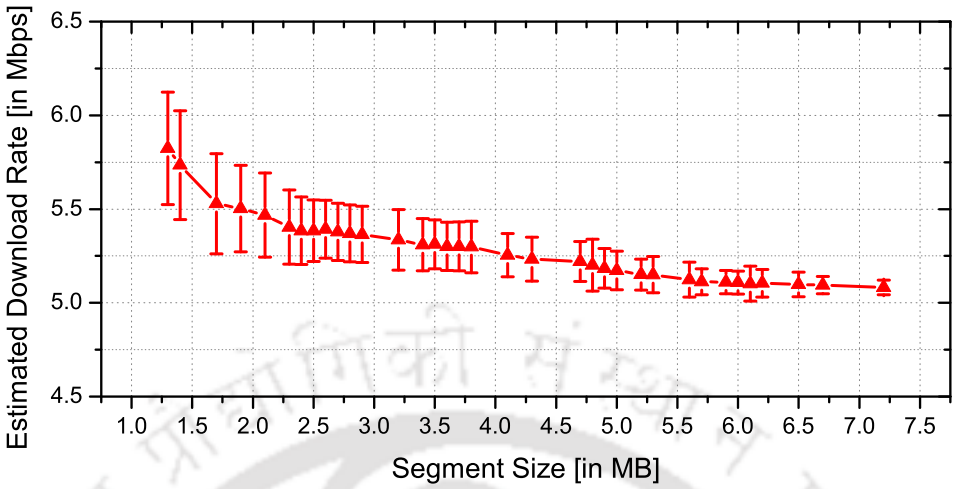


Figure 5.1: Impact of segment size variation on download rate estimation in DASH.

are pictorially illustrated in Fig. 5.2a, Fig. 5.2b and Fig. 5.2c, respectively. The mean segment download times in Fig. 5.2 are obtained from multiple testbed measurements with the client-AP link capacity set to 1 Mbps and packet loss ratio to 2%.

From Fig. 5.2a and Table 5.1, we can observe that the segment download time for all segments of Big Buck Bunny with 0.6 Mbps video bitrate ranges between 0.5 seconds to 1.5 seconds. On the contrary, the segment download time for all segments of Big Buck Bunny with 3.5 Mbps video bitrate varies between 2 seconds to 7 seconds (refer Fig. 5.2a). We have also observed similar variation in segment download time for the Tears of Steel video sequence, especially with 6.0 Mbps video bitrate. Fig. 5.2b demonstrate that for segments of Tears of Steel video sequence with 6.0 Mbps video bitrate (with maximum and minimum segment size (in KBytes) as 15000 and 2200, respectively), the segment download time significantly varies between 4 seconds to 14 seconds. The variation in the segment download time, especially at higher video bitrate levels, when streamed using a bandwidth throttled link is mostly attributed to the variation in segment sizes. Thus, by observing Fig. 5.2, we deduce that not all video segment responses have the same download time and could cause frequent playback buffer exhaustion.

An appropriate DASH/2 adaptation algorithm must estimate the time required to download the next κ segments before determining the video bitrate level for the next κ segments (which also determines the size) to avoid the playback buffer exhaustion. The DASH/2 K-push adaptation algorithms in the literature consider either the estimated

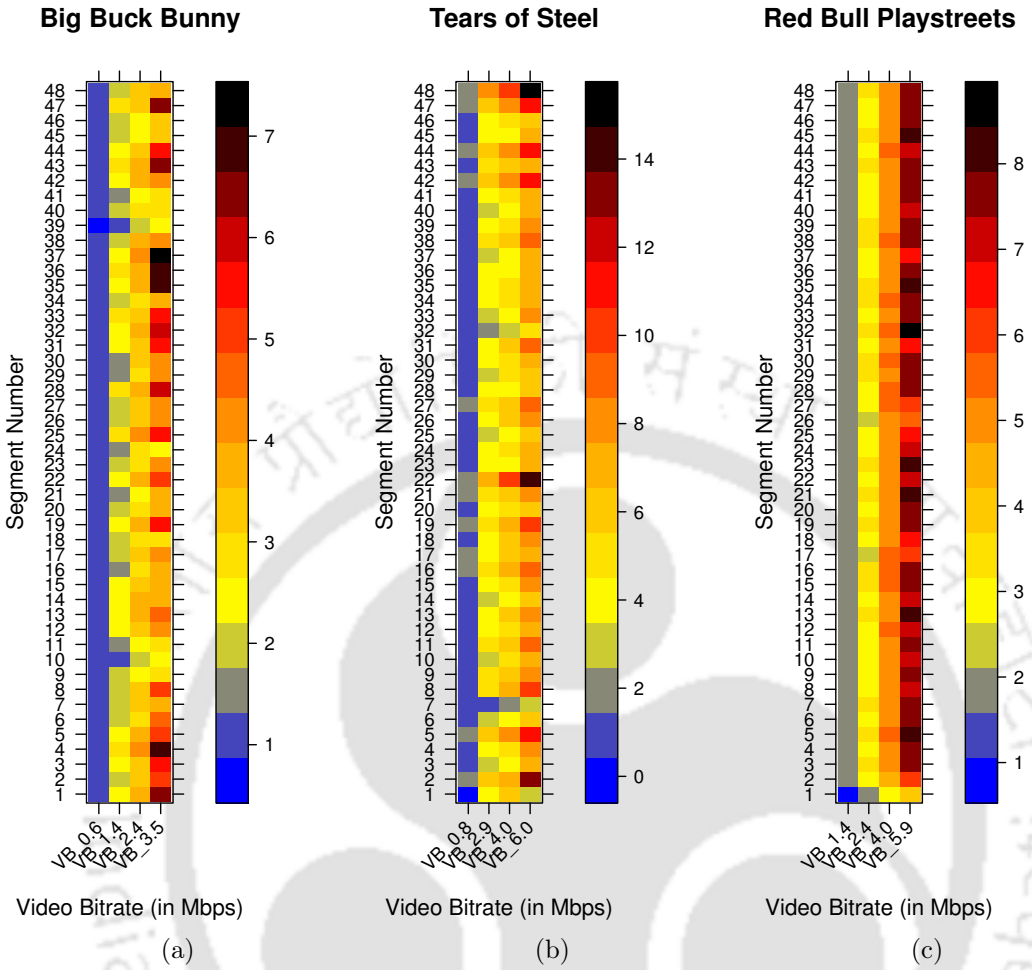


Figure 5.2: Observed variation in segment download times for different video bitrates with 10 second segment play duration of (a) Big Buck Bunny, (b) Tears of Steel, and (c) Red Bull Playstreets.

network bandwidth, playback buffer occupancy, or a combination of both to decide the suitable adaptation push pair for a push cycle. They do not take into account the variation in the segment size and its impact on the estimated throughput and/or buffer filling rate. Hence, they could fail to accurately forecast the time required to download the κ segments in a push cycle, particularly in mobile clients operating under varying bandwidth conditions. Moreover, this might results in frequent buffer depletions (resulting in interruptions) and selecting a lower video bitrate level for the segments in the subsequent push cycles.

Furthermore, the K-push schemes proposed in the research literature do not take into consideration the influencing QoE factors affecting QoE of a DASH/2 session, including

video bitrate quality, playback continuity, bitrate smoothness, and initial startup delay for performance evaluation. The authors in [28] performed a QoE assessment of bitrate adaptation algorithms and their results demonstrate mutually dependence among the major influencing QoE metrics. Thus, an adaptive K-push scheme with the goal to merely maximize a subset of the influencing QoE metrics does not necessarily lead to a smoother viewing experience for the end-user.

In this chapter, we propose ASK-Push, an adaptive segment-aware K-Push algorithm to determine the appropriate adaptation push pair (κ, $\hat{b}_\kappa$) for DASH/2 with an ultimate goal of improving QoE in mobile clients.

5.2 ASK-Push: Adaptive Segment-aware K-push Algorithm

In this section, we provide a detailed description of the proposed adaptive segment-aware K-push algorithm (termed ASK-Push) for making an appropriate adaptation push pair decision for the push cycle. In the rest of this section, we also discuss the ASK-Push associated components: enhanced MPD file and network bandwidth estimation.

5.2.1 Enhanced Media Presentation Description File

A typical Media Presentation Description (MPD) file in DASH is an XML document containing information about media segments such as periods (describe the content with a start time and duration), segment play duration, media representations, segment URIs, and other metadata that clients may require. The video segment can be a separate media file or a sub-range of a larger video file. The MPD can put segment byte-range and duration information directly in the Segment Index Box (sidx) of each track and specify the URI of sidx. However, the current DASH dataset does not include the segment size information in the MPD. Accordingly, we propose to enhance the standard MPD file by listing the segment sizes in the MPD file. This can be accomplished either during the MPD generation or the separate pre-processing phase. The enhance MPD file with the segment size information is then hosted at the DASH server. The segment size information is subsequently used in the throughput estimation module (in Section 5.2.2) as well as in the ASK-Push algorithm (in Section 5.2.3) to make an informed adaptation push pair decision.

5.2.2 Bandwidth Estimation

Accurate bandwidth estimation is vital for an adaptation algorithm while selecting the appropriate bitrate level for the segments in a push cycle. An average of the throughputs measured for all the segments downloaded in the current video session will provide a better insight in picking the suitable bitrate. The authors in [45], used *harmonic mean* to avoid the effect of instantaneous variation in the measured throughput and reduce the impact of outliers. Nonetheless, the average download rate over HTTP will have significant variation since the segments being downloaded are of different sizes, and it solely depends on the file size distribution [168]. Accordingly, we have used the *weighted harmonic mean* to estimate the bandwidth for the next segment to be downloaded. We have assigned weights $(W_1, W_2, \cdots)$ to the video segments wherein the weights are proportional to the respective segment sizes. The weighted harmonic mean download rate is calculated using the assigned weights and their respective segment download rates $(d_1, d_2, \cdots)$. The weighted harmonic mean download rate (HM_n) for n segments is defined by

$$HM_n = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{d_i}}, \tag{5.1}$$

using which the time to download the next segment is estimated to be W_{n+1}/HM_n.

5.2.3 Proposed ASK-Push Algorithm

The proposed ASK-push algorithm selects the number of segments and an appropriate video bitrate level from the set of available bitrates ($\mathbb{R}$) for the next push cycle to be initiated. After receiving the responses, the segments are placed in the playback buffer of maximum size B_{max}. We have also assumed three buffer thresholds: B_{res}, B_{crit} and B_{max}, as pictorially represented in Fig. 5.3, associated with the playback buffer. These three buffer thresholds are defined in terms of the number of segments (each with fixed play duration). We have also assumed that the threshold values will remain constant for the video session's entire duration. The notations used in the ASK-Push algorithms, together with their meaning, are listed in Table 5.2.

Requirements: The main objective of the proposed algorithm is to maximize the overall QoE by selecting the best possible adaptation push pair leveraging both the estimated bandwidth and current playback buffer occupancy. This simply translates to four requirements

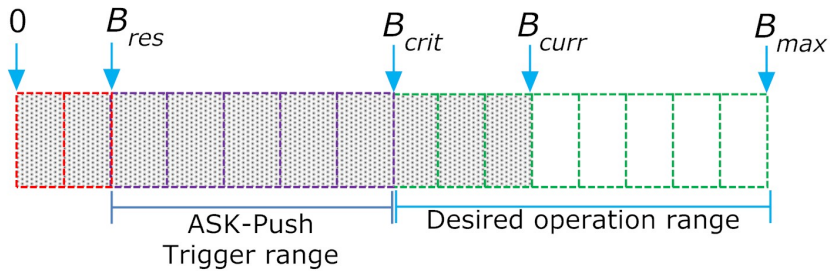


Figure 5.3: ASK-Push playback buffer model with different buffer thresholds.

Table 5.2: Notations used in ASK-Push algorithm.

Symbol	Description
$\mathbb{R}$	Set of available video bitrates
r^{min}	Lowest available video bitrate
r^{max}	Highest available video bitrate
φ	Segment play duration
B_{curr}	Current playback buffer occupancy at time t
B_{res}	Buffer reservoir for the fast start phase
B_{crit}	Critical buffer threshold
B_{max}	Maximum buffer threshold
K_{max}	Maximum number of video segments allowed to push during given push cycle; to avoid the over-push problem
κ	Actual number of segments selected for the push cycle
$\hat{b}_{\kappa}$	Video bitrate selected for a push cycle; same bitrate for all κ segments
n	Number of segments downloaded so far
HM_n	Weighted harmonic mean throughput estimated on n segment downloads
V_l	Total video session length
V_d	Video content downloaded so far (i.e., $n \times \varphi$)
g^*	Number of segments pushed (i.e., group size) during previous push cycle
b^*	Segment bitrate in the last push cycle

discussed below:

- **R1**: Minimize the playback startup delay to prevent viewers from abandoning the video session at an early stage [169]. Here, the algorithm cannot compute the weighted harmonic mean download rate to determine the adaptation push pair for the first cycle due to the absence of historical data for the download rate.

- **R2**: To ensure segments with higher video quality are streamed to the viewer, once sufficient playback buffer is accumulated (i.e., buffer occupancy goes beyond B_{res}), while avoiding the playback buffer depletion.

- **R3**: With favorable network conditions, the algorithm should avoid the superfluous filling of the buffer to limit the wastage of resources if the viewer quits the session prematurely.
- **R4**: The algorithm must react promptly when the mismatch between the estimated download time and the actual download time is discovered, which causes delayed segment delivery during the push cycle.

Minimizing Initial Startup Delay (R1): To address the **R1**, when the current buffer occupancy B_{curr} is below the buffer reservoir threshold B_{res}, the minimum number of segments required to fill the playback buffer beyond the buffer reservoir threshold (i.e., $B_{curr} \geq B_{res}$) is computed. Subsequently, the newly computed value is then assigned as the number of segments (κ) for the push cycle. In addition, the lowest video bitrate, from $\mathbb{R}$, is selected for fetching the κ segments in the push cycle. This not only ensures a smaller startup delay at the beginning of the video session but also makes sure that the buffer is filled as early as possible after a playback buffer depletion. Moreover, the aim of selecting the lowest possible bitrate for the segments of the initial push cycle is to avoid the weighted harmonic mean calculation.

Video Quality Maximization (R2): Once sufficient playback buffer is accumulated (i.e., buffer occupancy reaches beyond B_{res}), the ultimate objective of our proposed ASK-Push algorithm (i.e., **R2**) is of two-fold:

1. Improve the average video quality in order to achieve higher long-term user engagement [21].
2. Avoid the playback buffer depletions that lead to frequent playback interruptions and could result in a degraded end-user viewing experience.

Prior to initiating a push cycle request, the number of segments κ and their corresponding video bitrate $\hat{b}_\kappa$ are determined to attain the objectives as mentioned earlier. We call this phase as the *adaptive K-push phase* in the rest of the chapter. First, we proposed a buffer estimation function, presented in Algorithm 1, in order to estimate playback buffer occupancy after downloading κ video segments of bitrate $\hat{b}_\kappa$ in a push cycle. The buffer estimation function takes κ, $\hat{b}_\kappa$, B_{curr} and HM_n as input and returns the estimate of playback buffer occupancy after downloading the κ segments in a push cycle. At the beginning

of every push cycle, the anticipated buffer occupancy B_{esti} is estimated using the κ and $\hat{b}_\kappa$ values obtained from the previous push cycle. Based on the estimated B_{esti} value, the ASK-Push algorithm could follow either one of the bitrate adaptation strategies discussed below.

- **Uphill adaptation**: Uphill adaptation is performed when the buffer estimate B_{esti} value is greater than the buffer threshold B_{crit} and presented in Algorithm 2. The uphill adaptation aggressively increases both video bitrate $\hat{b}_\kappa$ and the number of segments κ of the adaptation push pair for the current push cycle. This ensures segments with higher video quality are streamed to the viewer. As shown in Fig. 5.3, the region between the two thresholds B_{crit} and B_{max}, is the most desired operation range for buffer occupancy B_{curr}. Based on the estimated bandwidth and buffer estimate B_{esti}, the uphill adaptation function gradually increases both the video bitrate level and the number of segments (in single steps) for the current push cycle. The step by step increment for both video bitrate level ($\hat{b}_\kappa$) and push cycle group size (κ) is repeated until the condition $B_{esti} < B_{crit}$ is met. The gradual increment of $\hat{b}_\kappa$ and κ by the uphill adaptation function not only ensures segments with an appropriate higher video quality are streamed to the viewer but also reduces the

ALGORITHM 1: Buffer estimation function for ASK-Push algorithm.

Data:
$W_{n+i} = \{w_{n+1}, w_{n+2}, \ \dots\ , w_{n+\kappa}\}$, are the segment sizes for κ segments with video bitrate $\hat{b}_\kappa$, respectively.

Input:
κ: Number of segments considered (group size)
$\hat{b}_\kappa$: Bitrate chosen for segments
n: Number of segment downloaded so far
HM_n: Weighted harmonic mean download rate for the last n segments
B_{curr}: Current playback buffer occupancy (in seconds)

Initialization:
$B_{esti} \leftarrow B_{curr}$;
$B_{esti} \leftarrow \max(0, B_{esti} - RTT/2)$; `// Time required to send the REQUEST massage`
for $i \leftarrow 1$ ***to*** κ **do**
  $B_{esti} \leftarrow B_{esti} + \varphi - \left(\frac{W_{n+i}^{\hat{b}_\kappa}}{HM_n}\right)$;

Result:
B_{esti}: The estimated buffer occupancy after downloading κ segments of bitrate $\hat{b}_\kappa$ in a push cycle.

unpleasant effect of swift bitrate switches with a larger amplitude. In addition, by providing equal priority to the push cycle group size (κ) and video bitrate level ($\hat{b}_\kappa$), the uphill adaptation function ensures a lower segment *request message overhead* for the entire video session. The step size in uphill adaptation is a configurable parameter and can be set based on the desired aggressiveness.

- **Downhill adaptation**: If the buffer estimate B_{esti} value is smaller than the buffer threshold B_{crit}, the control is then passed to downhill adaptation function, presented in Algorithm 3, to make an informed decision for adaptation push pair (κ, $\hat{b}_\kappa$). The downhill adaptation follows a conservative approach while making the decision for both video bitrate $\hat{b}_\kappa$ and an appropriate number of segments κ for the current push cycle. The goal of downhill adaptation is not to initiate a push cycle request with current bitrate, if the time estimated to download the κ segments (i.e., B_{esti}) is greater than $B_{curr} - B_{res}$. Thus, this avoids selecting the lowest video quality in the subsequent push cycles and unwanted playback buffer depletions (resulting in

ALGORITHM 2: Uphill adaptation function for ASK-Push algorithm.

Initialization:
temp_$\hat{b}_\kappa \leftarrow \hat{b}_\kappa$;
temp_$\kappa \leftarrow \kappa$;

do
  $\hat{b}_\kappa \leftarrow$ temp_$\hat{b}_\kappa$;
  $\kappa \leftarrow$ temp_κ;

  if *(temp_*$\hat{b}_\kappa < r^{max}$*)* **then**
    temp_$\hat{b}_\kappa$++;
  else
    break;
  $B_{esti} \leftarrow$ Estimate_Buf(temp_κ, temp_$\hat{b}_\kappa$, n, HM_n, B_{curr});
  if *(* $(B_{esti} \geq B_{crit})$ && $(temp\_\kappa < K_{max})$ *)* **then**
    temp_κ++;
  else
    break;
  $B_{esti} \leftarrow$ Estimate_Buf(temp_κ, temp_$\hat{b}_\kappa$, n, HM_n, B_{curr});
while $(B_{esti} \geq B_{crit})$

Result:
κ: Group size for the current push cycle.
$\hat{b}_\kappa$: Segment bitrate selected for the current push cycle.

interruptions). The downhill adaptation follows a conservative approach by giving higher priority to video quality over the group size in a push cycle, as evident in Algorithm 3. Accordingly, in the downhill adaptation function, the push cycle group size κ is gradually decreased till either of the constraints $\kappa < K_{min}$ and $B_{esti} < B_{crit}$ is satisfied. Thereafter, if the buffer estimate with newly decided κ and $\hat{b}_\kappa$ remains lower then the buffer threshold B_{crit} (i.e., $B_{esti} < B_{crit}$), the segment video bitrate is decreased in a linear fashion and most appropriate bitrate level is selected to meet the constraint $B_{esti} \geq B_{crit}$. The motive behind choosing the $B_{esti} \geq B_{crit}$ condition in the downhill adaptation function is to avoid the frequent push cycle requests.

Avoiding Over-push Problem (R3): Here, we propose an approach that simply delays

ALGORITHM 3: Downhill adaptation function for ASK-Push algorithm.

Initialization:
temp_$\hat{b}_\kappa \leftarrow \hat{b}_\kappa$;
temp_$\kappa \leftarrow \kappa$;

do
  $\kappa \leftarrow$ temp_κ;
  if *(temp_$\kappa > K_{min}$)* **then**
    temp_$\kappa --$;
  else
    break;
  $B_{esti} \leftarrow$ Estimate_Buf(temp_κ, temp_$\hat{b}_\kappa$, n, HM_n, B_{curr});
while *($B_{esti} < B_{crit}$)*

$B_{esti} \leftarrow$ Estimate_Buf(κ, temp_$\hat{b}_\kappa$, n, HM_n, B_{curr});

if *($B_{esti} < B_{crit}$)* **then**
  do
    $\hat{b}_\kappa \leftarrow$ temp_$\hat{b}_\kappa$;
    if *(temp_$\hat{b}_\kappa > r^{min}$)* **then**
      temp_$\hat{b}_\kappa --$;
    else
      break;
    $B_{esti} \leftarrow$ Estimate_Buf(κ, temp_$\hat{b}_\kappa$, n, HM_n, B_{curr});
  while *($B_{esti} < B_{crit}$)*

Result:
κ: Group size for the current push cycle.
$\hat{b}_\kappa$: Segment bitrate selected for the current push cycle.

the adaptation push pair computation for a push cycle to address **R3**. When the buffer occupancy B_{curr} goes beyond the buffer threshold B_{crit}, the request for computing an appropriate adaptation push pair of new push cycle is initiated only after waiting for a time period δ. The value of the δ can be calculated as given by Eq. (5.2). In other words, the request for computing an adaptation push pair will only get initiated when the current buffer occupancy falls below B_{crit}. Thus, it addresses the over-push problem and limits the wastage of resources used to fetch the video segments in case the viewer decides to quit watching the video prematurely.

$$\delta = B_{curr} - B_{crit} \tag{5.2}$$

Push Cycle Termination (R4): The proposed ASK-Push algorithm ensures the accuracy in predicting the segment download time by leveraging both weighted harmonic mean bandwidth prediction and segment size information. Thus, the ASK-Push algorithm selects the highest possible $\hat{b}_\kappa$ and appropriate κ, while making sure that the playback buffer occupancy B_{curr} is maintained in the desired operating range. Nevertheless, in reality, the bandwidth prediction can deviate and not always guaranteed, particularly when the client operates in a cellular network and the prediction period is sufficiently long. A mismatch can be discovered by periodically comparing the actual measured bandwidth with the predicted bandwidth value. The bandwidth mismatch can be quantified using the formula from [27] and is defined as

$$BW_{Diff}(t) = \int_{t_0}^{t} BW_r(t)dt \; - \; \int_{t_0}^{t} BW_p(t)dt, \tag{5.3}$$

where $BW_r(t)$ and $BW_p(t)$ are real measured bandwidth value and predicted bandwidth value, respectively. The first component represents actual consumed network resources, and the second component represents the predicted consumed network resources starting from the beginning of the push cycle (i.e., t_0). If the measured real bandwidth value is sufficiently higher than the predicted one, the delivery of video segments with the expected video quality is guaranteed. On the contrary, the risk of interruption or selecting the lower video bitrate level in the subsequent push cycles arises due to the delayed segment delivery. Hence, it could degrade the overall quality of experience of the current streaming session.

To address the challenge as mentioned in **R4**, we propose push cycle termination making

use of the HTTP/2 stream termination feature. Once the ASK-Push approach discovers bandwidth mismatch $BW_{Diff}(t)$ that reaches beyond a threshold, it subsequently cancels out all the upcoming pushed segments by leveraging the stream termination feature. The complete push cycle termination works as follows. The client initiates the push cycle request with a *push directive*. Afterward, the client receives a *PUSH_PROMISE* frame notifying the video segments that are being pushed by the server. The client extracts the segment URIs and their corresponding stream IDs by parsing the received *PUSH_PROMISE* frame and preserved for later usage. The client hands over the pushed segments to the application level as soon as it receives all the content in the promised streams. If a bandwidth mismatch beyond the threshold is discovered, the client cancels all the promised but not completely received streams by sending *RST_STREAM* frames associated with the stream IDs.

Putting It All Together: We now describe the overall flow of the ASK-Push algorithm, presented in Algorithm. 4. It starts when the client needs to determine the number of segments and appropriate bitrate level (or representation) for the push cycle according to the current network conditions. The ASK-Push algorithm is initialized with the set of available video bitrates ($\mathbb{R}$: $\{r^{min}, ..., r^{i}, ..., r^{max}\}$) and the three buffer thresholds: B_{res} (default=2), B_{crit}, and B_{max}. The values of these parameters remain constant for the entire video session. Note that each time the ASK-Push algorithm is called with the following input: recently downloaded segment number (say n; 0 if none), bitrate level of the segment(s) in the most recent push cycle (b^*; default is set to r^{min} for the very first push cycle), number of segments pushed in the most recent push cycle (g^*), current playback occupancy (b_{curr}), the sizes of $(n+1)^{th}$ segment across all the available video bitrate levels (i.e., $w^{min}_{n+1}, ..., w^{i}_{n+1}, ..., w^{max}_{n+1}$), and the estimated weighted harmonic mean download rate (HM_n).

In the very first push cycle or after a playback interruption (due to playback buffer depletion), the κ value for the push cycle is computed as the minimum number of segments required to fill the playback buffer such that the playback buffer occupancy reaches beyond the buffer reservoir threshold (i.e., $B_{curr} \geq B_{res}$). Subsequently, the lowest bitrate level r^{min}, from $\mathbb{R}$, is selected for fetching the κ video segments of the push cycle. The delay factor δ value is set to zero in order to initiate the push cycle request immediately and fill the buffer rapidly. The adaptation enters in the *adaptive K-push phase*, once the buffer occupancy

ALGORITHM 4: ASK-Push algorithm for DASH/2

Data:
$\mathbb{R} = \{r^{min}, \ldots, r^{i}, \ldots, r^{max}\}$, Set of available video bitrates
B_{res}, B_{crit}, B_{max}: Buffer thresholds (in seconds)

Input:
n: Number of segment downloaded so far
b^*: Segment bitrate chosen in the most recent push cycle
g^*: # of segments pushed in the most recent push cycle
B_{curr}: Current playback buffer occupancy (in seconds)
HM_n: Weighted harmonic mean download rate for the last n segments
$W_{n+1} = \{w^{min}_{n+1}, .., w^{i}_{n+1}, .., w^{max}_{n+1}\}$, are the (n+1)th segment sizes for video bitrates $\{r^{min}, ..., r^{i}, ..., r^{max}\}$, respectively.

Initialization:
if $(B_{curr} \leq B_{res})$ **then**
  $K_{max} \leftarrow \left(\left\lceil \frac{B_{res}-B_{curr}}{\varphi} \right\rceil + 1\right)$;
  $\kappa \leftarrow K_{max}$;
  $\hat{b}_{\kappa} \leftarrow r^{min}$;
  $g^* \leftarrow \kappa$;
  $b^* \leftarrow \hat{b}_{\kappa}$;
  $\delta \leftarrow 0$;
else
  if $(B_{curr} \leq B_{crit})$ `// Adaptive K-push`
  then
    $K_{max} \leftarrow min\left(\left\lfloor \frac{(B_{max}-B_{curr})}{\varphi} \right\rfloor, \left\lceil \frac{(V_l-V_d)}{\varphi} \right\rceil\right)$;
    $K_{min} \leftarrow \left(\left\lceil \frac{(B_{crit}-B_{curr})}{\varphi} \right\rceil + 1\right)$;
    $\kappa \leftarrow max(g^*, K_{min})$;
    $\hat{b}_{\kappa} \leftarrow b^*$;
    $B_{esti} \leftarrow$ Estimate_Buf(κ, $\hat{b}_{\kappa}$, n, HM_n, B_{curr});
    if $(B_{esti} \geq B_{crit})$
    then
      Bitrate_Uphill_Adaptation(κ, $\hat{b}_{\kappa}$, K_{max});
    else
      Bitrate_Downhill_Adaptation(κ, $\hat{b}_{\kappa}$, K_{min});
    $g^* \leftarrow \kappa$;
    $b^* \leftarrow \hat{b}_{\kappa}$;
    $\delta \leftarrow 0$;
  else
    `// Avoiding Over-push Problem`
    $\delta \leftarrow (B_{curr} - B_{crit})$;

Result:
κ: Group size for the current push cycle.
$\hat{b}_{\kappa}$: Video bitrate selected for the κ segments.
δ: The wait time before deciding appropriate κ and $\hat{b}_{\kappa}$ for the next segment push cycle.

reaches beyond the buffer threshold B_{res}.

During the *adaptive K-push phase*, initially the maximum number of segments (K_{max}) and the minimum number of segments (K_{min}) for the current push cycle is computed to avoid the over-push problem and fill the buffer beyond the B_{crit} threshold, respectively. The appropriate K_{max} is computed by leveraging both the current buffer occupancy and total video length. Subsequently, the K_{max} is used to cap the κ either to minimize or even eliminate the over-push problem. The group size κ for the current push cycle is initialized with the maximum of either K_{min} or g^*. Thereupon, the video bitrate $\hat{b}_\kappa$ is initialized with b^*, where b^* is the video bitrate selected during the last push cycle. The estimate of playback buffer occupancy B_{esti} is then computed with κ and $\hat{b}_\kappa$ using Algorithm 1. Thereafter, the newly computed buffer estimate B_{esti} is compared with the buffer threshold B_{crit}. If the buffer estimate B_{esti} value is greater than the buffer threshold B_{crit}, the uphill adaptation function thereupon decides the appropriate adaptation push pair (κ, $\hat{b}_\kappa$) for the current push cycle, as described in Algorithm 2. Otherwise, the control is passed to the downhill adaptation function, as described in Algorithm 3, to make an informed decision for κ, and $\hat{b}_\kappa$ of the current push cycle.

Every time the B_{curr} goes bellow B_{crit}, the algorithm enters into the ASK-Push request trigger range (as shown in Fig. 5.3). Thus, ASK-Push starts to be more optimistic and tries to maintain the best possible video quality while avoiding unwanted interruptions and reducing segment request overhead. In the *adaptive K-push phase*, the push cycle request is sent immediately without any delay.

Furthermore, with favorable network conditions, the playback buffer could get filled superfluously and leads to the over-push problem. To address this, the ASK-Push algorithm eventually waits until the buffer occupancy falls to B_{crit} before initiating a new push cycle request. The waiting time (i.e., δ) is computed as the difference between current buffer occupancy B_{curr} and the buffer threshold B_{crit}. Thus, the delayed download limits the total number of video segments in the player buffer and avoids filling the playback buffer superfluously. In addition, this limits the wastage of resources in case the user quits watching the video prematurely.

Illustration with Scenario-based Example: In this subsection, an example of the overall approach in practice is provided for better comprehension of the proposed ASK-Push

scheme. This scenario is given in accordance with the proposed ASK-Push algorithm, presented in Algorithm. 4. In our scenario, we assume that the buffer model parameters configured with values as $B_{res} = 2$, $B_{crit} = 6$, and $B_{max} = 12$ segments. We have also considered the total video session duration as 400 seconds with a segment play duration of 10 seconds.

At the beginning of the video session, the goal is to minimize initial startup delay (R1), or after an interruption due to playback buffer depletion, the goal is to resume the playback at the earliest possible. In either of the situations, the current buffer occupancy B_{curr} is below the buffer reservoir threshold B_{res}. Hence, the minimum number of video segments required to fill the playback buffer beyond the buffer reservoir threshold is computed as $\left(\left\lceil \frac{B_{res}-B_{curr}}{\varphi} \right\rceil + 1\right)$. The computed value, i.e., 3, is then assigned to the maximum number of segments allowed for the push cycle, K_{max}. Subsequently, the value K_{max} is directly selected as the number of segments for the push cycle, κ. Next $\hat{b}_{\kappa}$, the video bitrate selected for a push cycle (same bitrate for all κ segments), is initialized with the minimum available video bitrate, r^{min} to fill the buffer as early as possible. Hence, in our case, for the initial push cycle, the κ and $\hat{b}_{\kappa}$ are assigned with three and r^{min}, respectively. Now, the g^* and b^* parameters are updated with the κ and $\hat{b}_{\kappa}$ values, respectively and the same will be utilized in the subsequent iteration. The delay factor δ is set to zero in order to dispatch the push cycle request immediately. If all the segments in the previous push cycle are downloaded successfully, the B_{curr} will go beyond the buffer reservoir threshold B_{res}, i.e., $B_{curr} \geq 20$ seconds.

The algorithm will enter into the adaptive K-push phase in the second iteration and merely focus on video quality maximization (R2). The K_{max} value is computed as the minimum between 9 and 37 and K_{min} is assigned with value 4. Next, the κ value in the current iteration is initialized with the maximum value observed between K_{min} and g^*. Thus, the value of κ in the current iteration is set to 4. The $\hat{b}_{\kappa}$ value for the current iterations is then initialized with b^* (i.e., r^{min}). Subsequently, the ASK-Push algorithm determines the estimated buffer occupancy after downloading four segments of bitrate equals to r^{min} in a push cycle. The estimated buffer occupancy computation also requires the weighted harmonic mean download rate for the last n segments. Based on the estimated buffer occupancy B_{esti} value with respect to the B_{crit} value, the ASK-Push algorithm either performs uphill

adaptation or downhill adaptation. In uphill adaptation, the algorithm aggressively increases both $\hat{b}_\kappa$ and κ values in a stepwise manner till the condition $B_{esti} \geq B_{crit}$ is satisfied. On the contrary, in downhill adaptation, the algorithm conservatively first decreases the κ value till it reaches 4 or $B_{esti} \geq B_{crit}$ satisfied. Subsequently, the fresh estimated buffer occupancy is determined using $\hat{b}_\kappa$ and recently updated κ values. If $B_{esti} < B_{crit}$, then the $\hat{b}_\kappa$ value decreased in a linear fashion to meet the constraint $B_{esti} \geq B_{crit}$. In every iteration the algorithm checks for the condition $B_{curr} > B_{crit}$. If found true, the delay factor δ is set with the $B_{curr} - B_{crit}$ value to avoid the Over-push Problem. In our case, assume the current buffer occupancy, B_{curr} value is 80 seconds. Then, the delay factor δ value is set with 20 seconds. Thus, no further push cycle requests will be initiated for the next 20 seconds.

Complexity Analysis: In general, using a larger number of video bitrate levels is considered to be better to achieve finer adaptation for DASH. The search time for the ASK-Push algorithm to decide an appropriate video bitrate level for the segments in a push cycle increases linearly with an increase in the number of available bitrate levels of the DASH video. This is mostly because the ASK-Push algorithm to address the Video Quality Maximization (R2) requirement (includes both uphill adaptation and downhill adaptation) estimates the time required to download the appropriate κ segments for all the bitrate levels in every push cycle. Accordingly, the ASK-Push algorithm follows an aggressive and conservative approach to select the most suitable video bitrate, $\hat{b}_\kappa$, with a goal to avoid the buffer depletion for uphill adaptation and downhill adaptation, respectively. Hence, the ASK-Push algorithm has a worst-case time complexity in $\mathcal{O}(|\mathbb{R}|)$, where $|\mathbb{R}|$ denotes the cardinality of the set of available bitrate levels of the DASH video. The linear time complexity of our ASK-Push algorithm also makes it very amenable for implementation in mobile devices.

5.3 DASH/2 System Implementation

In this section, we present the implementation details of a DASH/2 streaming prototype used to conduct experiments on an emulated network testbed with real-world network traces. To support the adaptive-push mechanism in a DASH/2 system, both the streaming server and the client should understand the push protocol. Accordingly, our DASH/2 streaming prototype consists of three key components: an HTTP/2 streaming server, a streaming

client, and a network traffic shaper for network emulation. The complete architecture of the prototype, together with the different modules implemented in both the client and streaming server, is illustrated in Fig. 5.4. In the rest of the section, we present a brief description of the system components.

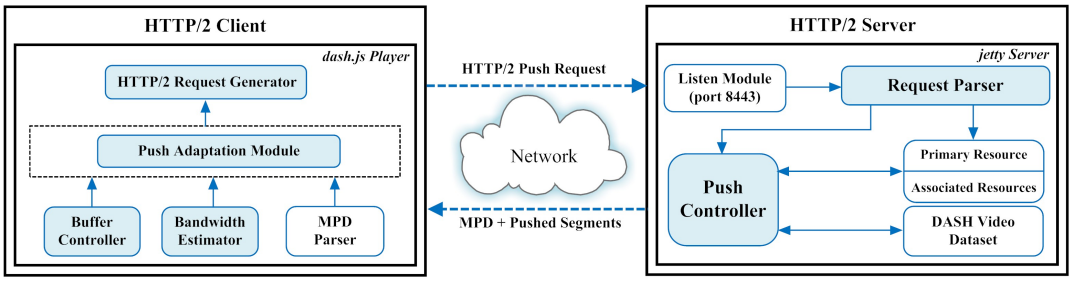


Figure 5.4: DASH/2 system architecture together with the modules implemented in both streaming server and streaming client.

5.3.1 DASH/2 Streaming Server

We implement our DASH/2 streaming server in Linux/Unix platform. We chose the Jetty (release 9.4.11.v20180605) [38], a Java-based open-source HTTP server implementation, to build our HTTP/2 streaming server because of its HTTP/2 support. The following major functions were implemented on the streaming server-side. We have activated additional SSL and HTTP/2 modules in the server to add HTTPS and HTTP2 support for the server. The server side also contains a listen module with support for SSL/TLS encrypted connection. We have configured the HTTP/2 push mechanism by adding a *PushCacheFilter* filter-mapping in the web.xml file. Nevertheless, the automated HTTP/2 web push would not be able to perform the desired K-Push functionality. To address this, we have implemented two custom modules namely; *Request Parser* and *Push Controller* as a *Filter* class (as shown in Fig. 5.4). The streaming server passes the HTTP/2 request, received on the listen module, to the *Request Parser* module. The *Request Parser* module identifies the *PushDirective* header field in the received HTTP/2 request. It then parses the adaptation push pair $(\kappa, \hat{b}_\kappa)$ and their segment byte ranges (when the segment is a sub-range of a larger file). Afterward, the *Request Parser* identifies the primary resource URI and constructs an associated push resources array leveraging the received adaptation push pair. Subsequently,

the *Push Controller* module in the streaming server will perform the following: 1) build a new stream to send the response for the primary resource, 2) send the *PUSH_PROMISE* frames of the pushed segments over this stream, and 3) determine the dependencies of these pushed segments based on their play location and open a stream for each of the associated (pushed) segments within the newly launched thread.

5.3.2 DASH/2 Streaming Client

We have implemented the streaming client in the *dash.js* framework [35], an open-source JavaScript-based and potentially browser-independent DASH-compliant media player implementation. Our client implementation is based on the *dash.js* master branch (v1.2.0 release) as it was the stable version at the time of development. We have extended the MPD parser module to parse and record the segment size information for future usage. The original rule-based bitrate adaptation logic in the adaptation module is replaced with our proposed ASK-Push algorithm[1]. This module is usually triggered when the playback buffer level goes below the critical buffer threshold (i.e., B_{crit}). The proposed ASK-Push logic employed in the adaptation module determines the appropriate adaptation push pair (κ, $\hat{b}_\kappa$) for the push cycle. Afterward, the client sends the adaptation push pair (κ, $\hat{b}_\kappa$) to the streaming server, as an HTTP/2 request message with a *PushDirective* field, for launching the push cycle. The module also gets triggered to initiate the push cycle termination request if a bandwidth mismatch beyond a specified threshold is discovered.

5.3.3 Network Traffic Shaper

We have used a simple traffic shaping script called Wondershaper [150], that uses Linux *TC* tool in the background to shape the bandwidth of the client-server link. The *TC* uses a token bucket filter to throttle the bandwidth of a link. We have written a Python script to process the raw trace files (as described in Section 5.4.2) and extract the throughput values and their corresponding time duration from the trace files. The network shaper throttles both the uplink and downlink capacity with the throughput values obtained during the pre-processing stage of the respective trace files. The packet-loss rate is set to 0% because it is already included in the bandwidth traces' fluctuations. The network topology of our

[1] `https://github.com/hemakyarnagula/ASK-PushAlgorithmCode/`

trace-driven experimental testbed is depicted in Fig. 5.5. In Fig. 5.5, local indicates the link with effectively unbounded bandwidth and lower access latency.

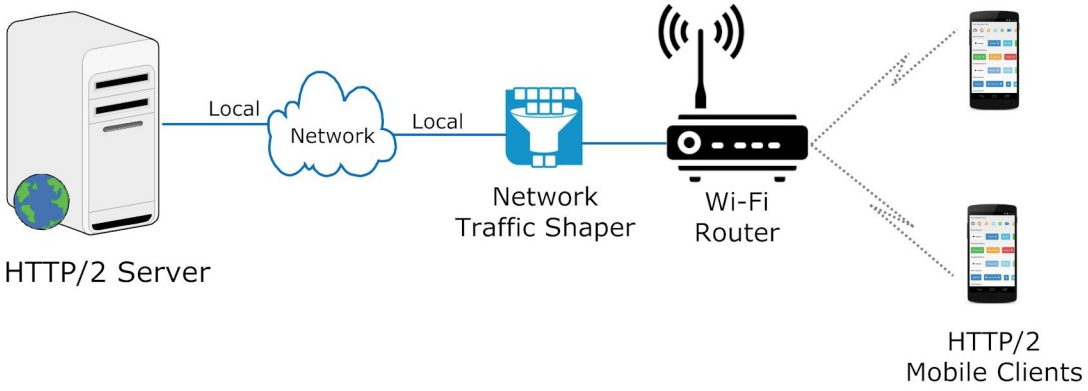


Figure 5.5: Network topology configured in the DASH/2 trace-driven experimental testbed.

5.4 Performance Evaluation

In this section, we evaluate our ASK-Push scheme by conducting controlled experiments on an emulated real mobile trace-driven network testbed. We also discuss the details of the experimental setup, DASH video dataset, mobile bandwidth traces and QoE metrics used in the video QoE evaluation.

5.4.1 Experiment Testbed Setup

We have implemented our DASH/2 streaming prototype in Linux/Unix platform. Our experimental testbed consists of four nodes: HTTP/2 streaming server, network traffic shaper with TC controller, Wi-Fi router (configured with 802.11n), and mobile client(s). The HTTP/2 streaming server was deployed on a Linux machine with a 64-bit Intel Pentium CPU 2.8 GHz dual-core, 4 GB memory. The installed operating system is Ubuntu 12.04 with Linux kernel 3.13.0-66-generic. The network traffic shaper with a TC controller is installed in the HTTP/2 server to manipulate the network bandwidth. The Samsung i9500 Galaxy S4 smartphone, with octa-core CPU and 2GB RAM, running Android 4.2.2 OS was used for the client.

5.4.2 Mobile Bandwidth Traces

The performance of video quality assessment is closely correlated with the underlying network bandwidth conditions. Unlike in wired networks, mobile networks are mostly characterized by their intense throughput variations. Additionally, the prolonged low bandwidth due to diverse coverage quality could lead to an increased probability for interruptions. To perform the quality assessment under challenging bandwidth fluctuations as observed in a real-world network with mobility, we have used the throughput traces from HSDPA dataset [151, 152]. The throughput was measured in Telenor's 3G/HSDPA mobile network across popular commute routes in Norway. We considered three throughput profiles correspond to direct throughput measurements from a train, car and ferry, respectively. The trace consisted of 30 minutes of contiguous 1 second application-layer throughput measurements recorded with a video streaming client. These traces also had sufficient variation in their average throughput and throughput profile due to the mobility characteristics (i.e., the speed at which the device is traveling).

5.4.3 DASH Video Dataset

To perform the video quality assessment, we have selected two representative open movie sequences from the DASH dataset [153]. The first movie is Big Buck Bunny (BBB), a high-motion computer-animated movie. The second is Red Bull PlayStreet (RBPS), which

Table 5.3: Summary of the two test video sequences' resolutions and encoding rate (in Mbps) used for video quality assessment in a smartphone with 5 inch Full HD Super AMOLED display, 1920 x 1080 pixels resolution and 16:9 ratio (441 ppi density).

RI	BBB		RBPS	
	MER	Resolution	MER	Resolution
1	0.045	320x240	0.10	320x240
2	0.177	480x360	0.25	480x360
3	0.317	480x360	0.39	480x360
4	0.569	854x480	0.89	854x480
5	0.987	1280x720	1.50	1280x720
6	1.40	1280x720	2.50	1280x720
7	2.40	1920x1080	4.00	1920x1080
8	3.50	1920x1080	5.90	1920x1080

MER: Maximum encoding rate (in Mbps)
RI: Representation Index

is a sports documentary with high-motion scenes. We have considered eight different representations/bitrate levels for the two test video sequences. The video resolution and an encoding rate of the bitrate levels for the two video sequences are summarized in Table 5.3. The number of bitrate levels is consistent with the bitrate levels used by YouTube [155]. Additionally, considering a moderate number of distinct bitrate levels would lead to a very small probability of frequent bitrate switching events during the video playback. The total video length was restricted to about 480 seconds and consisted of 48 segments with a segment duration of 10 second each. The segment duration (i.e., 10 seconds) is consistent with the range of chunk durations that are widely used in real-world commercial services [156]. Although segments with different play durations could be considered, we found that this factor does not affect the video QoE assessment behavior.

5.4.4 DASH/2 Adaptation Algorithms

For performance comparison, in addition to our proposed ASK-Push algorithm, we implemented two recent DASH/2 push-based adaptation schemes from the literature. The summary of all the DASH/2 push-based adaptation schemes used for comparison are as follows.

1. **Adaptive-push (A-push):** We employed the *adaptive-push* function suggested by Xiao et al. [26] that dynamically decides the κ for the next push cycle. The function iteratively invoked to decide the appropriate value of κ using the predicted bandwidth and buffer length. The function also slows down or stops the increment of κ using two thresholds to address the over-push problem.

2. **DASH2M:** The DASH2M [27] scheme calculates the overall energy requirement of a push cycle. The energy requirements for a push cycle typically depend on the type of device, network type, client distance from the base station, etc. The DASH2M determines the overall energy consumption for a push cycle in a smartphone using the power profile provided in GreenTube [170]. Subsequently, the appropriate number of segments for the push cycle is determined that leads to higher energy efficiency and ensures continuous video playback. Once the group size (κ) is decided, DASH2M formulates the segment bitrate selection as an *integer linear programming* problem

with several constraints. The video bitrate for the segments in the push cycle is obtained by solving the *integer linear programming* problem. Note that we do not consider the server support for predicting the available bandwidth variation in our implementation. Our DASH2M implementation solely relies on the client-side local measured information for the prediction.

3. **ASK-Push:** Our proposed approach aims not only to improve video quality and reduce request overhead but also avoid playback buffer depletions using appropriate buffer thresholds. For our ASK-Push implementation, the client buffer model was configured with parameter values as $B_{res} = 2$, $B_{crit} = 6$, and $B_{max} = 12$ segments. These thresholds decide the size of the playback buffer (in seconds). The choice of different threshold values that include $B_{res} = 2$, $B_{crit} = 6$, and $B_{max} = 12$ segments were based on extensive experimental evaluation and observations on the algorithm performance in terms of objective QoE metrics. The choice of threshold values for ASK-Push implementation was studied with different buffer thresholds, under varying network bandwidths in the similar lines of some previous studies reported in [39]. For brevity, we do not provide more details of the extensive experimental evaluation and the observations on the algorithm performance in terms of objective QoE metrics; instead, we refer readers to previous studies reported in [39]. Thus, to configure the client buffer model with parameter values, we simply select the appropriate buffer thresholds from our observations that reduce request overhead and avoid playback buffer depletions.

5.4.5 Evaluation Objective QoE Metrics

The key requirements in QoE assessment are gathering, aggregation, and correlation of the measurements, in order to assess overall QoE. The application-layer objective QoE metrics can be effectively obtained by leveraging ubiquitous client-side QoE measurements. In light of these advantages, we have considered a set of six objective QoE metrics for the video quality evaluation (as discussed in Chapter 3.5.1). For completeness, we enumerate the set of objective QoE metrics considered for performance evaluation.

1) Bitrate Adaptation Efficiency (BAE)
2) Interruption Ratio (IR)

3) Average Interruption Duration (AID)

4) Bitrate Switching Amplitude Reward (BSAR)

5) Video Continuity Index (VCI)

6) Initial Startup Delay Reward (ISDR)

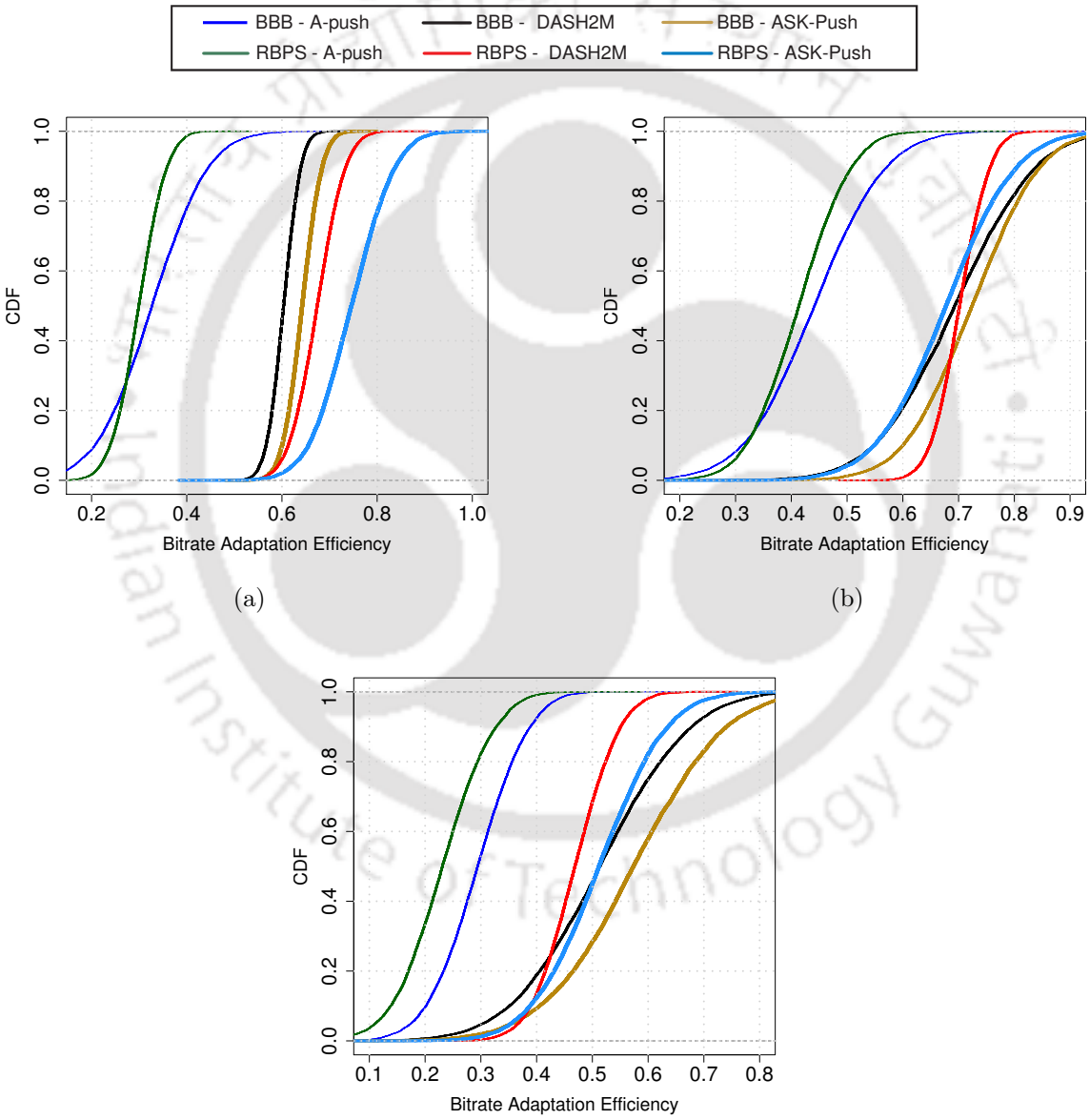


Figure 5.6: CDF of bitrate adaptation efficiency with different traces: (a) Train (b) Car and (c) Ferry.

5.4.6 Results and Discussions

We report the results obtained from experiments done with the two video sequences, which are streamed using the three bandwidth trace profiles. We use the QoE metrics discussed earlier to compare the performance of the three adaptation algorithms (discussed in Section 5.4.4). Fig. 5.6 shows the CDF of the bitrate adaptation efficiency for the three algorithms across the three bandwidth trace profiles. We observe that our proposed ASK-Push algorithm leads to higher BAE because the bitrate for the push cycle in the ASK-Push algorithm is determined purely leveraging both the current buffer occupancy and segment size information. As the buffer level gets beyond the threshold B_{res}, the ASK-Push algorithm aggressively selects higher representation for the segments in a push cycle until the estimated buffer level B_{esti} stays beyond the critical buffer threshold B_{crit}. The goal in determining the segment bitrate aggressively prior to the κ selection is to stream a higher video quality as it has a greater influence on the end-user viewing experience. On the contrary, DASH2M is conservative in both κ and $\hat{b}_\kappa$ selection. It solves the integer linear programming problem to select the bitrate for each segment of a push cycle with constraints such as maximum buffer size and continuous playback in order to stabilize the buffer occupancy and avoid interruptions, respectively. As a result of this, it has recorded lower BAE as compared to ASK-Push across all the trace-video scenarios. The A-push algorithm fares poorly under all the scenarios due to the use of instantaneous bandwidth measurement for both the group size and segment bitrate selection. The lower BAE for the A-push scheme, across all experimental scenarios, can be attributed to the inaccurate bandwidth estimation by the client under the time-varying network conditions. Both the ASK-Push and DASH2M schemes have recorded significantly lower BAE when streamed with the ferry trace profile, as shown in Fig. 5.6c. The ferry trace has several sharp drops in the bandwidth, lasting for a longer duration. Thus, it is evident that the ASK-Push and DASH2M adapt to the underlying network conditions by leveraging the bandwidth prediction and push cycle termination feature.

Fig. 5.7 shows the IR observed for the algorithms across different scenarios. From Fig. 5.7, it is also evident that the ASK-Push algorithm outperforms the others in terms of the IR. The ASK-Push algorithm leads to very few interruptions (i.e., lower IR) across all the scenarios since it considers segment size information both for bandwidth estimation

and segment bitrate selection. Moreover, the ASK-Push algorithm utilizes the segment size information to estimate the expected buffer occupancy at the end of a cycle before initiating the push cycle. Thus, it makes sure the player buffer level stays beyond the critical buffer threshold B_{crit} as much as possible. However, the time-varying bandwidth scenarios could lead to playback buffer depletions. The ASK-Push scheme initiates the push cycle termination request when a bandwidth mismatch is discovered. Once the push cycle termination is invoked, a fresh push cycle request with appropriate κ and $\hat{b}_\kappa$ is sent. If the network condition further deteriorates, this could lead to buffer depletion until the segment response from the freshly initiated push cycle arrives. From Fig. 5.7 and Table 5.4, we can observe that the ASK-Push has fewer short duration interruptions across all the scenarios.

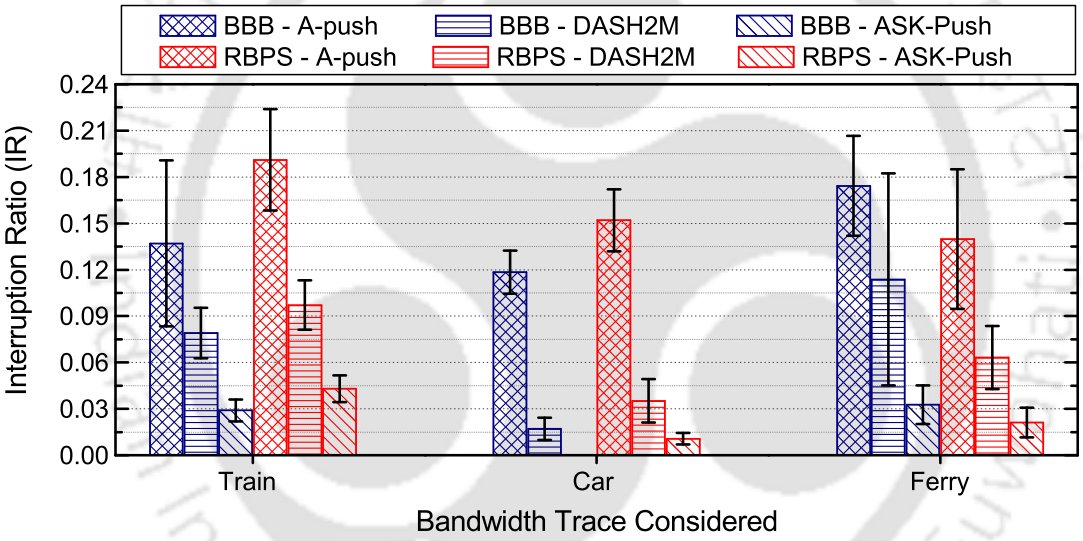


Figure 5.7: Interruption ratio with different throughput traces: Train, Car and Ferry.

On the other hand, higher IR (i.e., more number of interruptions) is observed for the A-push scheme under all the experiment scenarios. The A-push scheme determines the group size and video bitrate based on the predicted bandwidth and segment duration only. Unlike ASK-Push or DASH2M, it neither discovers the bandwidth mismatch nor uses the stream termination feature to reset the current push cycle streams. As a result, it suffers from frequent interruptions (resulting in a higher IR) under time-varying bandwidth conditions. It is also evident from the values reported in Table 5.4 that A-push has interruptions of longer duration (resulting in a higher average interruption duration), especially with train and ferry traces. Nonetheless, A-push has comparatively fewer interruptions of moderate duration

under the car trace scenario. The interruptions of moderate duration for the A-push can be attributed to the higher bandwidth and fewer bandwidth drops in the car trace profile. We see from Fig. 5.7 that the DASH2M algorithm has moderate IR and also AID of few seconds as reported in Table 5.4. Although the DASH2M utilizes the stream termination feature, it does not take the segment size information into account while determining the adaptation push pair. This had resulted in higher push cycle terminations when several bandwidth mismatches were discovered. Furthermore, if the new push cycle response fails to reach the client before the buffer gets exhausted, this will usually lead to interruptions of mostly a smaller duration. A similar trend has been observed for DASH2M across all the scenarios.

The VCI observed for all the algorithms across different scenarios is plotted in Fig. 5.8. The ASK-Push algorithm outperforms all others across the scenarios. The higher VCI for the ASK-Push algorithm is attributed to its efficacy in avoiding the unnecessary interruptions during the video session by appropriately determining the adaptation push pair. The VCI for DASH2M is marginally lower than ASK-Push across all scenarios. This is mostly due to the comparatively higher IR (i.e., more number of interruptions) observed for DASH2M. Although the AID for DASH2M is lower (as reported in Table 5.4), the cumulative pause time is significantly high due to the higher IR, which in turn results in lower VCI for DASH2M. We have also observed a lower VCI for the A-push algorithm under both train and ferry scenarios, as shown in Fig. 5.8. The lower VCI is mainly attributed to interruptions of longer duration (also resulting in a larger AID).

The BSAR values observed for the three algorithms across the experiments are plotted in Fig. 5.9. In all the scenarios, the ASK-Push algorithm has outperformed the others in terms of BSAR. This is mostly due to the aggressive increase of the video bitrate in subsequent push cycles and maintaining a stable buffer reservoir to avoid bitrate switches with higher amplitude. The measured ISDR values for all the algorithms while streaming with the three trace profiles are reported in Table 5.4. We notice that all the algorithms have similar ISDR values in all the scenarios. This is due to the aggressive fetching of segments with the lowest bitrate (during the very first push cycle) to fill the buffer reservoir as early as possible and start the session.

Table 5.5 summarizes the overall improvements (in percentage) by the ASK-Push algorithm when compared with the other approaches. We observe that the ASK-Push

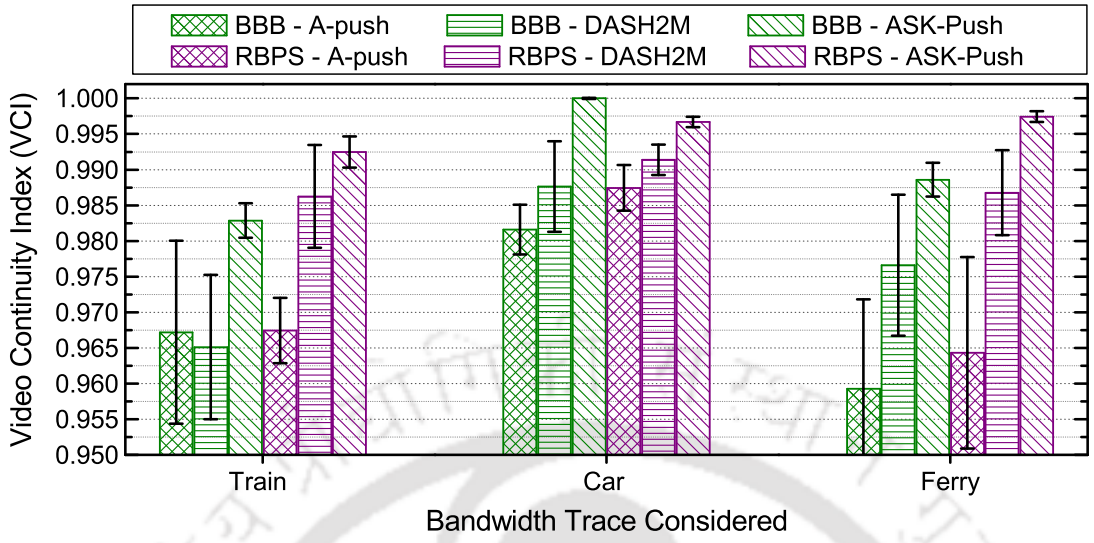


Figure 5.8: Video continuity index with different throughput traces: Train, Car and Ferry.

algorithm outperforms the others in terms of the six objective QoE metrics across the trace-video scenarios. We can observed that BAE is increased by 6.46%, BSAR is improved by 6.51%, IR is decreased by 63.29%, and AID is reduced by 78.16% (resulting in an improvement in VCI) when the BBB video sequence is streamed with train trace (as shown in Table 5.5). In Table 5.5, the proposed ASK-Push algorithm has significantly reduced the IR (as reported in column 4) as it follows a conservative approach to maintain sufficient

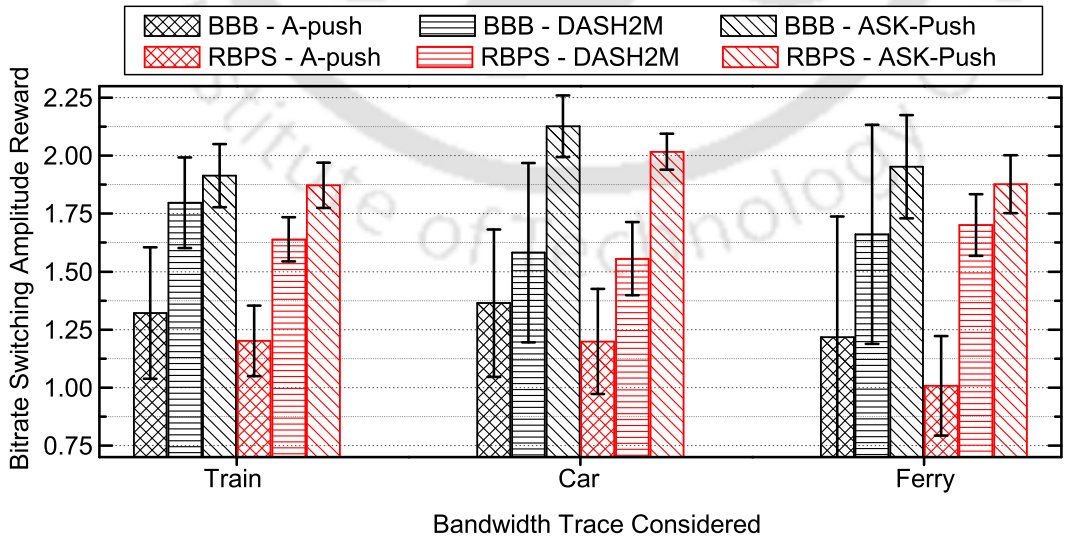


Figure 5.9: Measured bitrate switching amplitude reward with three bandwidth trace profiles: Train, Car and Ferry.

buffer level and avoids the buffer depletions. When the RBPS video sequence is streamed with the car trace, we also observe that the BAE is decreased by 3.5%. Nevertheless, the ASK-Push algorithm has demonstrated significant improvements for both IR and AID (i.e., 69% and 71%) in the RBPS-car trace scenario. This is attributed to the conservative nature of the ASK-Push algorithm that tries to maintain a stable buffer level leveraging the segment size information. From Table 5.5, we have observed significant improvements for all the objective QoE metrics with the ASK-Push algorithm in all the other scenarios as well.

Table 5.4: Average interruption duration (in seconds) and initial startup delay reward with different trace-algorithm scenarios.

Video Seq	Metric	Trace	Adaptation Algorithms		
			A-push	DASH2M	ASK-Push
Big Buck Bunny	AID	Train	11.245 (2.55)	2.354 (1.26)	0.514 (0.03)
		Car	6.241 (0.15)	0.814 (0.63)	0.0 (0.0)
		Ferry	17.654 (6.51)	5.457 (1.96)	1.487 (0.66)
	ISDR	Train	0.0	0.03133	0.05060
		Car	0.17349	0.14458	0.18795
		Ferry	0.08193	0.15181	0.18554
Red Bull Playstreets	AID	Train	13.548 (3.65)	1.954 (0.71)	0.631 (0.08)
		Car	2.124 (1.02)	0.871 (0.33)	0.249 (0.05)
		Ferry	12.658 (5.33)	2.954 (1.32)	1.294 (0.45)
	ISDR	Train	0.15897	0.18145	0.23589
		Car	0.0	0.06585	0.17985
		Ferry	0.18746	0.23589	0.18261

Note: Values represent mean, with stdev for AID in brackets.

Table 5.5: Improvements in objective QoE metrics (in percentage) across different experiment scenarios.

Video	Trace	QoE Metrics					
		BAE	IR	AID	BSAR	VCI	ISDR
BBB	Train	6.46	63.29	78.16	6.51	3.37	61.51
	Car	4.49	100	100	34.45	1.25	8.33
	Ferry	11.46	71.25	72.75	17.58	0.95	22.22
RBPS	Train	10.69	55.72	67.71	14.22	0.63	30.0
	Car	-3.48	69.8	71.41	29.63	0.54	173.1
	Ferry	4.64	66.4	56.19	10.35	1.08	22.59

Table 5.6: QoE score predicted with parametric MLR model across different scenarios.

Video Sequence	Trace	Adaptation Algorithms		
		A-push	DASH2M	ASK-Push
Big Buck Bunny	Train	3.0667 (0.22890)	3.7333 (0.26469)	4.2333 (0.22403)
	Car	3.2333 (0.22403)	3.9667 (0.23928)	4.2667 (0.20872)
	Ferry	3.1667 (0.23178)	4.0333 (0.23928)	4.1333 (0.22501)
Red Bull Playstreets	Train	2.8667 (0.22501)	3.5333 (0.26133)	4.0667 (0.24744)
	Car	3.4333 (0.26048)	4.0667 (0.24744)	4.3667 (0.22005)
	Ferry	3.2667 (0.22890)	4.1333 (0.22501)	4.2334 (0.20337)

Note: Values represent mean QoE score, with CI in brackets.

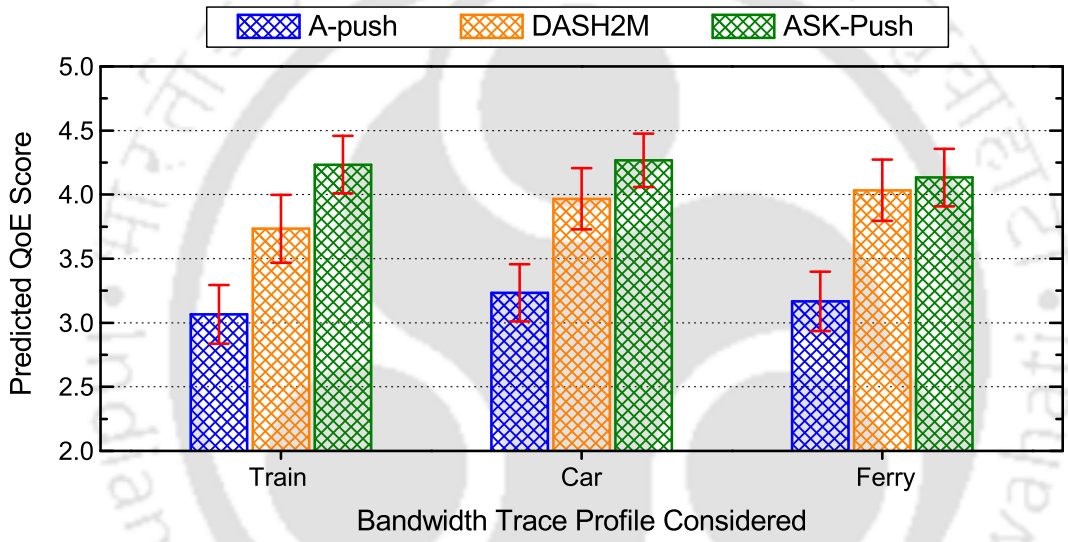


Figure 5.10: Illustration of QoE score with confidence interval predicted by the parametric MLR model for DASH/2 adaptation algorithms while streaming Big Buck Bunny video sequence.

QoE Score Estimation: In a bid to measure the overall performance of the adaptation algorithms, we have provided the values measured for objective QoE metrics as input to the parametric MLR model (as discussed in Chapter 4) and estimated the QoE score. The QoE score predicted by the MLR model is reported in Table 5.6, where the values represent the mean of the predicted QoE score with confidence interval in brackets. Furthermore, we have plotted the QoE score predicted with parametric MLR model across different scenarios (reported in Table 5.6) to visually observe if the confidence intervals of the different adaptation algorithms overlap with each other. The graphs illustrating QoE score with confidence interval across different trace scenarios while streaming with Big Buck Bunny

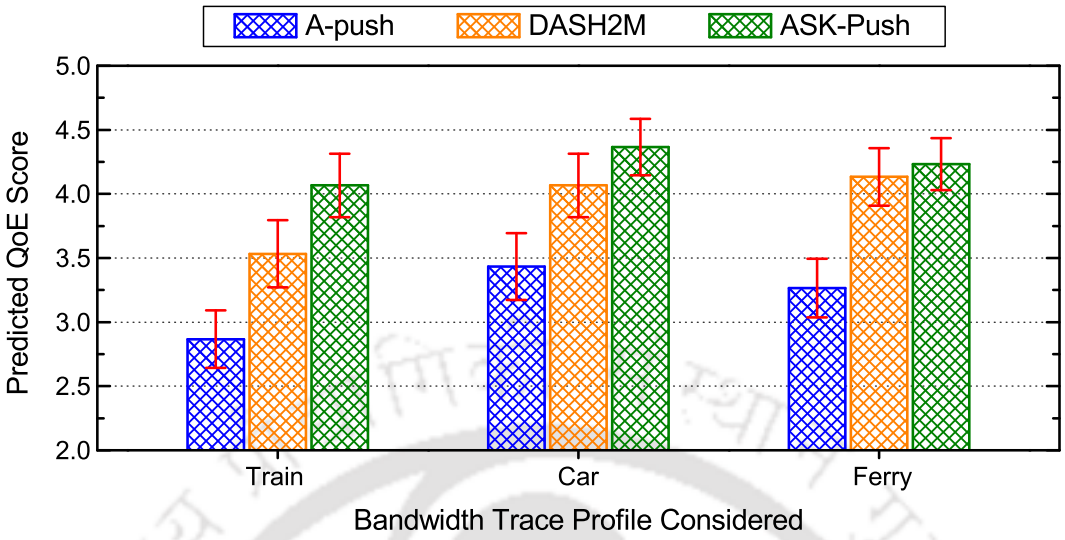


Figure 5.11: Illustration of QoE score with confidence interval predicted by the parametric MLR model for DASH/2 adaptation algorithms while streaming Red Bull Playstreets video sequence.

and Red Bull Playstreets video sequence are shown in Fig. 5.10 and Fig. 5.11, respectively. It can be observed from Table 5.6 that our proposed ASK-Push algorithm outperforms the others in terms of the predicted QoE score across all the video-trace scenarios.

5.5 Summary

In this chapter, we proposed an adaptive segment-aware K-push (ASK-Push) algorithm for DASH/2 streaming. The ASK-Push algorithm leverages the potential of the segment size information to determine an appropriate adaptation push pair for the next push cycle on runtime. A buffer control model with three buffer thresholds is employed to maintain the buffer occupancy between the dual-threshold with a goal to avoid both buffer underflow and overflow. Our proposed ASK-Push algorithm also utilizes an upper buffer threshold to cap the group size of the push cycle to eliminate the DASH/2 over-push problem. We have implemented a DASH/2 streaming prototype and the ASK-Push algorithm in an open-source `dash.js` framework. We validated our proposed ASK-Push algorithm by conducting extensive experiments on an emulated network testbed with real-world mobile network traces. We have used real mobile devices and a broad set of emulated mobile network conditions to assess the QoE improvements. Our experimental results have demonstrated

the efficiency of the ASK-Push algorithm, especially with time-varying mobile network conditions. The experimental results have demonstrated that the ASK-Push algorithm achieved an average of 6%, 67% and 19% improvement for bitrate adaptation efficiency, interruption ratio and bitrate switching amplitude reward, respectively, when compared to the other approaches. Furthermore, QoE metrics measurements have confirmed that the proposed approach achieved playback smoothness while maintaining a significantly lower interruption ratio and a higher video continuity index. The ASK-Push algorithm also outperforms the others in terms of the overall QoE score predicted using the parametric MLR model across all the scenarios considered.

CHAPTER

6

An Adaptive Segment Prefetching Strategy for QoE-Improved DASH in Multi-Access Edge Computing Networks

The previous chapter presented an adaptive segment-aware K-push algorithm that has focused on improving the QoE for DASH/2. The adaptive segment-aware K-push algorithm successfully mitigates the request explosion problem in mobile streaming clients while significantly improving the end-user QoE for DASH sessions. This chapter aims to provide a different perspective for improving the end-user QoE for DASH by considering a segment prefetching strategy that can bring the video segments closer to the streaming client while reducing the client-video source access latency in Multi-access Edge Computing (MEC) networks. To this end, this chapter presents an adaptive segment prefetching strategy in MEC networks to dynamically decide the number of segments and prefetch those segments with an appropriate video bitrate ahead of the client request progress. The gains brought by the proposed segment prefetching strategy have been confirmed through an emulated MEC testbed implementation. Particularly, extensive experimentation shows that the proposed adaptive segment prefetching approach outperforms in terms of prefetched segment hit ratio and bitrate adaptation efficiency when compared to baseline schemes under various video streaming scenarios.

Organization of the Chapter: The remainder of this chapter is organized as follows. We first present the motivation for the work in Section 6.1. Section 6.2 provides the full description of the proposed adaptive segment prefetching strategy. The experimental testbed

implementation details, baseline comparison schemes, and performance metrics are described in Section 6.3. We also analyze the performance of the proposed ASPF approach from the scenario-based experiment results, which are discussed in Section 6.3. Finally, we draw conclusions in Section 6.4.

6.1 Motivation

The *Multi-access Edge Computing* (MEC), a promising alternative for streaming video segments directly from the network edge, was introduced by the standards organization European Telecommunications Standards Institute (ETSI) in December 2014 [31]. The MEC paradigm enables both mobile network operators (MNO) and multimedia content providers to directly collaborate at the mobile network edge (as discussed in Chapter 2.4). Of late, several research works have proposed segment prefetching at the network edge for improving end-user video QoE [32, 33, 34] in the literature. The works in [32, 33] have proposed proxy-based prefetching schemes that can be deployed at the wireless access points. Nevertheless, these existing prefetching schemes (as discussed in Chapter 2.5.3) follow a rigid policy, i.e., only prefetch either one video segment at a time or a fixed number of segments ahead of time. The DASH client typically decides the video segment bitrate based on the dynamic network conditions and playback buffer occupancy prior to sending the segment request. Thus, prefetching of a fixed number of video segments (of a particular video bitrate) beforehand does not perform well, especially in the radio access network (RAN) environment with highly fluctuating network conditions. This could also result in frequent segment misses at the network edge. In addition, prefetching of a fixed number of video segments could also lead to underutilization of the limited storage resources of the MEC server, especially if the prefetched video segments are stored for a longer duration before getting evicted due to a segment miss.

In this chapter, we propose an adaptive segment prefetching (termed ASPF) strategy to dynamically determine the number of segments and their corresponding video bitrate for prefetching with an aim to improve the end-user QoE in mobile clients. The proposed strategy predicts and prefetches the appropriate number of video segments leveraging the last-mile radio link throughput prediction and virtual client buffer at the MEC server while maintaining a sufficient gap ahead of the client's actual segment request progress.

In a nutshell, our proposed ASPF scheme works as an enhanced proxy implementing the opportune prefetching of video segments on a per-user per-session basis at the MEC server.

6.2 Proposed Adaptive Segment Prefetching Strategy

In this section, we present an adaptive segment prefetching (ASPF) Strategy that is able to potentially support QoE-improved DASH delivery from original video sources that are located far away from the clients. The adaptive segment prefetching Strategy guarantees that video segments of appropriate video bitrate are available at the MEC server storage before receiving the clients' segment requests. For an adaptive video streaming session, the MEC server is responsible for handling all the segment requests from the clients. If a video segment requested by the client is available at the MEC server, it is directly served to the client with low access latency as the video segment is already located at the network edge. In case the requested segment is unavailable at the MEC server, the MEC server simply forwards the segment request to the source video server. Subsequently, the MEC server retrieves the requested video segment(s) and serves them immediately to the clients. In addition,

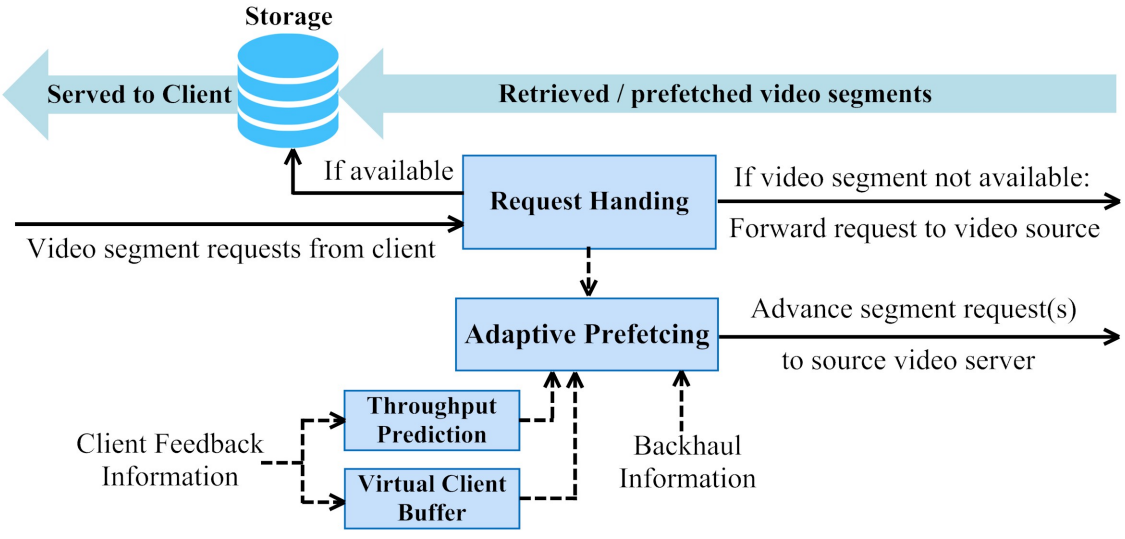


Figure 6.1: A high-level functional block diagram of MEC server with segment prefetching capabilities.

the content intelligence employed by the content provider at the MEC server performs the segment prefetching operation by downloading the segments well in advance, i.e., ahead of the client's actual request progress. Note that the MNO does not participate in any content operations at the MEC server. As the entities performing the content operations at the MEC platform are owned on a rented basis by the content/service providers along with the video source servers. Therefore, the content operations that include video segment retrieval and prefetching all the way between the MEC server and the original video streaming server can be applied to encrypted content. In Fig. 6.1, we present a high-level overview of the functionalities of the MEC server with segment prefetching capability in more detail. Furthermore, apart from the client request handling function as described earlier, the MEC server also has the functionalities of 1) context monitoring and throughput prediction (discussed in Section 6.2.1), 2) virtual client buffer simulation (discussed in Section 6.2.2), 3) adaptive segment prefetching, and 4) video quality adaptation for prefetched segments (discussed in Section 6.2.3).

Otherwise stated, the proposed prefetching scheme will address the issues that arise due to the bottleneck links in the end-to-end video delivery process, even when the client-video source path is cross-continent across the global Internet. In addition, for popular video sequences, the prefetched segments can be cached at the MEC server storage. In turn, these cached segments can be used for serving the subsequent segment requests from the other users. Unlike in the CDN caches where the CDN providers manage them, the cache at the MEC server can be flexibly managed by the video content providers with their own content intelligence (i.e., caching policies). In this work, we focus on adaptive segment prefetching at the network edge for DASH applications, while the MEC-based caching feature for DASH applications is out of this work's scope.

In the rest of the section, we present the description of the two key enabling modules for the ASPF scheme, including 1) RAN link throughput prediction and 2) virtual client buffer inference during the video sessions, together with the adaptive prefetching decision engine functionality. The two modules provide necessary inputs to the adaptive prefetching decision engine of the ASPF scheme. The adaptive prefetching decision engine makes an informed decision for prefetching the segments at the MEC server, leveraging the inputs from the two modules as mentioned earlier.

6.2.1 RAN Link Throughput Prediction Mechanism

The MEC server is aware of real-time Radio Access Network context information, i.e., information about real-time download throughput perceived by each of the streaming clients, without any feedback signaling from the clients. The real-time context information (especially the RAN conditions) is periodically disseminated through the MNO-owned Radio Network Information Service (RNIS) module at the MEC server [31]. The RNIS module is able to provide real-time context knowledge leveraging an interaction with low-level API at clients and the eNodeB. Based on the historic per-segment download throughput information collected, the proposed throughput prediction mechanism builds throughput prediction models that can able to predict the future throughput on the RAN link. The throughput prediction models can be utilized by the proposed ASPF scheme implemented at the MEC server to make decisions on video quality adaptation for segment prefetching while efficiently handling client requests. Alternatively, the proposed throughput prediction mechanism could leverage the Common Media Server Data (CMSD) [171] proposal to acquire the per-segment download throughput information at the MEC Server as perceived by the streaming client.

Several prior works in the research literature have used a wide range of simple prediction models for throughput prediction, mostly based on the past observations during the streaming session [32, 45, 68, 172]. The simple prediction models include but are not limited to Finite-state Markov channel (FSMC), Auto-Regressive model (AR), Last-Sample (LS) and Harmonic-Mean (HM). In FSMC, the channel states are represented by maximum supported bandwidth, and the transition probabilities are derived using statistical methods. The AR is a classical time series-based transition modeling technique that only uses past data to model/predict future behavior. The LS merely uses the data from the last observation for throughput prediction. The HM uses the harmonic mean of past throughput measurements in the video session for throughput prediction.

Of late, the authors in [89] have investigated the performance of these simple prediction models in the wild for both initial and midstream throughput prediction during a video session. Their analysis on throughput variability using a proprietary dataset of throughput measurements from a leading online video content provider uncovers several insightful observations. The key observations from their throughput variability analysis [89] include:

Observation 1: There is exists a significant amount of throughput variability within a video session.

Observation 2: The evolution of the throughput within a video session exhibits various stateful/persistent characteristics. Thus, capturing these characteristics meaningfully and using a cross-session prediction methodology could lead to an improved throughput prediction.

Observation 3: The video sessions with a similar set of features at the network layer tend to exhibit the same initial throughput conditions and similar patterns for midstream throughput evolution.

Thus, simple statistical models are not expressive enough to capture the significant diversity in video session characteristics and are inaccurate in last-mile throughput prediction. To this end, Hidden Markov Model (HMM) which is an efficient state-based model has been widely used in the research literature to predict the last-mile link throughput and traffic properties [89, 173, 174, 175].

For our work, we consider a simple but effective throughput predictor based on HMM that is similar to the one in CS2P [89]. Our HMM-based throughput predictor is based on the first-order Markovian assumption, which essentially means that the next state (i.e., the future state) is dependent only upon the current state of the network link and is independent of any past states of the network link. The states in our HMM-based throughput predictor represents the measured throughput of the network link. The HMM-based throughput predictor utilizes three parameters for predicting the link throughput, namely (i) state transition probabilities (i.e., probability of moving from one hidden state to another), (ii) observation emission probabilities (i.e., probability of emitting a particular observation in a state), and (iii) initial state probability distribution (i.e., the initial probabilities of being in some state). The HMM-based throughput predictor captures the state-transition behaviour in each cluster $Agg(M_s^*, s)$. The M_s denotes a given set of features M from all possible feature combinations for any session s. The M_s^* denotes the set of features that are mapped to each session s and yields the lowest absolute normalized prediction error. The set of all such previous video sessions are aggregated and denoted by $Agg(M_s^*, s)$. Note that the session clustering steps used in our throughput predictor are similar to that in the

practical prediction system [176], wherein the optimal session clusters are chosen based on throughput prediction accuracy. For more details on the session clustering process, we refer the interested readers to [176].

In what follows, we start by formally defining the Hidden Markov Model used for predicting the throughput. Let W_t be the random variable representing the network throughput at time period t, and the actual downlink throughput measured from the RNIS module is denoted as w_t. $\hat{W}_t$ be the predicted value of W_t. We also assume that the network throughput W_t evolves according to some random hidden state variables $X_t \in \mathcal{X}$, where $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$. The variable $\mathcal{X}$ denotes the set of all possible discrete states and the number of states is given by $N = |\mathcal{X}|$. Taking into consideration that state X_t is a random variable, we represent the probability distribution for state X_t as a vector. The probability distribution vector for state X_t is denoted as:

$$\Pi = (\mathbb{P}(X_t = x_1), \mathbb{P}(X_t = x_2), \ \cdots \ , \ \mathbb{P}(X_t = x_N)). \tag{6.1}$$

The key assumption in the HMM with a first order Markovian assumption is that the state evolves as a Markov process where the probability distribution of the current state only depends on the state of the previous time period. Thus, we can write it as $\mathbb{P}(X_t|X_{t-1}, X_{t-2}, \cdots, X_1) = \mathbb{P}(X_t|X_{t-1})$. In addition, the transition probability matrix is then denoted by $\mathbf{P} = \{\mathbf{P}_{ij}\}$, where $\mathbf{P}_{ij} = \mathbb{P}(X_t = x_i|X_{t-1} = x_j)$. According to the Markov property, we can represent it as:

$$\Pi_{t+\tau} = \Pi_t \mathbf{P}^\tau. \tag{6.2}$$

Taking into consideration the hidden state X_t, we assume the probability distribution function (PDF) of throughput W_t (also called the emission PDF) is Gaussian:

$$W_t|X_t = x \sim N(\mu_x, \sigma_x^2) \ . \tag{6.3}$$

In general, the HMM could work with any emission PDF other than Gaussian. Note that we use Gaussian emission because of its high prediction accuracy and its computational simplicity [89]. Each state in the HMM follows a Gaussian distribution of network throughput. The Gaussian distribution of network throughput is denoted by the mean of the distribution and its standard deviation $N(\mu_x, \sigma_x^2)$. The transition probability is then computed between

every pair of states. Fig. 6.2 provides an example of a high-level overview of the HMM in our context. In summary, the HMM captures the state transitions and the dependency between the network throughput vs. the hidden state in order to predict the future throughput efficiently.

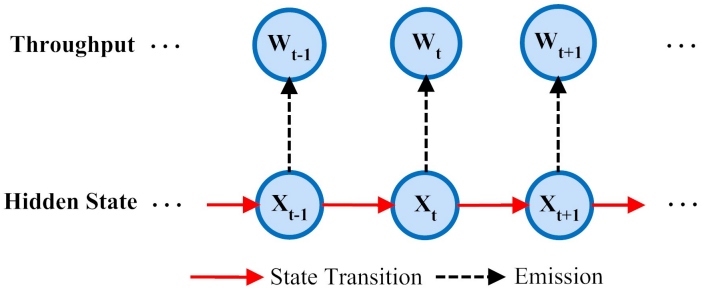


Figure 6.2: A high-level overview of the Hidden Markov Model.

Next, we introduce several notations that we will use for describing both offline training and online throughput prediction. For simplicity, we use $W_{1:t} = \{W_1, W_2, \cdots, W_t\}$ to denote the network throughput from time period 1 to time period t. Let $\Pi_{t_1|1:t_0} = (\mathbb{P}(X_{t_1} = x_1|W_{1:t_0}),\ \mathbb{P}(X_{t_1} = x_2|W_{1:t_0}),\ \cdots,\ \mathbb{P}(X_{t_1} = x_N|W_{1:t_0}))$ be the pdf vector of the random hidden state X_{t_1}, given the network throughput from time period 1 to t_0. For example, $\Pi_{t|1:t-1}$ represents the state of X_t given the network throughput up to time period $(t-1)$.

HMM Offline Training: For any given number of states, say N, we can use the training data in $Agg(M_s^*, s)$ to learn the parameters of HMM as: $\Theta_{HMM} = \{\Pi_0, \mathbf{P}, \{(\mu_x, \sigma_x^2), x \in \mathcal{X}\}\}$. The HMM parameters are estimated by the offline training procedure via a special case of the standard expectation-maximization (EM) algorithm (i.e., Baum-Welch algorithm) [177].

HMM Online Throughput Prediction: In this phase, a new session is mapped to the most similar session in the training dataset, which matches most of (if not all) the features with the session under prediction. Then, we use the corresponding HMM of that mapped session to make the throughput predictions. The HMM-based online throughput prediction algorithm is presented in Algorithm 5. At a high level, Algorithm 5 involves both throughput prediction for the next time period using HMM as well as updating the HMM state based on the actual measured throughput (which is provided by the RNIS module as feedback

ALGORITHM 5: Online prediction using HMM-based throughput predictor

Initialization:
Let t be the time period id.
for $i \leftarrow 1$ **to** T **do**
  if $t = 1$ *(initial time period)* **then**
    Initialize Π_1;
    $\hat{W}_1 = Median(Agg(M_s^*, s))$
  else
    $\Pi_{t|1:t-1} = \Pi_{t-1|1:t-1}\mathbf{P}$
    $\hat{W}_t = \mu_x, \quad \text{where } x = arg\max_{x \in \mathcal{X}} \mathbb{P}(X_t = x | W_{1:t-1}))$
  Provide prediction $\hat{W}_t$ to prefetching decision engine.
  Obtain actual throughput measurement w_t.
  Update $\Pi_{t|1:t} = \frac{\Pi_{t|1:t-1} \circ e(w_t)}{(\Pi_{t|1:t-1} \circ e(w_t)) \cdot \mathbf{1}}$.

signal). In what follows, we discuss the three key steps in the online throughput prediction approach:

1. **Throughput prediction (initial time period)**: The HMM predicts the initial throughput of a session s simply as the median throughput of sessions in $Agg(M_s^*, s)$, i.e.,

$$\hat{W}_1 = Median(Agg(M_s^*, s)). \tag{6.4}$$

   The median is simply considered as no historical throughput information for the current session is available at the beginning of the session. Note that our proposed prefetching approach does not require any throughput prediction at the beginning of each session. This is because the initial few video segment requests are resolved by the MEC server and directly download the video segments from the source video server.

2. **Throughput prediction (midstream time period)**: At time period t, given the updated probability distribution function of HMM state $\Pi_{t-1|1:t-1}$, the trained model can compute the state probability distribution function at current time period according to the Markov property (as discussed in Eq. 6.2):

$$\Pi_{t|1:t-1} = \Pi_{t-1|1:t-1}\mathbf{P}. \tag{6.5}$$

   The network throughput prediction $\hat{W}_t$ for time t is then computed via the maximum

likelihood estimate (MLE) and is given below:

$$\hat{W}_t = \mu_x, \quad x = arg\max_{x \in \mathcal{X}} \mathbb{P}(X_t = x | W_{1:t-1})). \tag{6.6}$$

3. **Updating HMM**: Once the actual client perceived downlink throughput information, w_t is received through the RNIS module; the information is used to update the state of HMM Π_t. Thus, this reflects the most updated information about the network condition. The HMM computes $\Pi_{t|1:t}$, given the actual downlink throughput $W_t = w_t$ and $\Pi_{t|1:t-1}$ using the equations stated below.

$$\Pi_{t|1:t} = \frac{\Pi_{t|1:t-1} \circ e(w_t)}{(\Pi_{t|1:t-1} \circ e(w_t)) \cdot \mathbf{1}}, \tag{6.7}$$

where $e(w_t)$ represents the emission probability vector and is defined as $e(w_t) = (f(w_t|X_t = x_1), f(w_t|X_t = x_2), \cdots, f(w_t|X_t = x_n))$. The $f(\cdot)$ is the Gaussian probability distribution function, $\circ$ denotes entry-wise multiplication of the two vectors (also called as Hadamard product).

Subsequently, the HMM-based throughput predictor determines the future downlink throughput, which is fed into the prefetching decision engine to make an informed prefetching decision for the video session. In addition, the MEC server is aware of the backhaul network conditions between itself and the video source, especially in terms of bandwidth, latency and packet loss. The MEC server regularly performs ping-like measurements in order to estimate the network condition between itself and the video source (whose IP address is already known). This information is leveraged by the prefetching decision engine at the MEC server to estimate the time required to download/prefetch the video segments from the video source. Thus, it will allow the proposed ASPF scheme to optimally schedule the prefetching segment requests.

6.2.2 Virtual Client Buffer Inference Mechanism

The DASH clients do not provide the client playback buffer status to the MEC server, which provides important inputs to the adaptive segment prefetching scheme. To the best of our knowledge, none of the existing DASH clients has implemented any buffer reporting

mechanisms yet. In contrast to the existing solutions, we design an approach where the MEC server infers the streaming clients' playback buffer status. Note that by "inference", we mean that such a mechanism that simulates the virtual client playback buffer (referred to as a mirrored buffer) at the MEC server leveraging the explicit feedback signaling from the client.

The objective of simulating the virtual client buffer at the MEC server is to avoid the client's delayed feedback while making an appropriate run-time decision by the prefetching decision engine as far as possible. To maintain an accurate estimate of the actual client buffer, we perform the end-to-end synchronization by periodically sending the clients' actual playback buffer state as feedback appended along with the segment request(s) of the client. To perform the end-to-end synchronization, we apply the concept of the Program Clock Reference in Moving Picture Experts Group (MPEG) systems [178].

The end-to-end synchronization can also be efficiently performed using the CTA-5004: Common Media Client Data (CMCD) specification [179] by the Consumer Technology Association (CTA) published in September 2020. To achieve the end-to-end synchronization, a proof-of-concept system with CMCD-aware `dash.js` client [180] using the CTA-5004 specification can directly convey certain information to the content delivery network servers with object requests. The CMCD-aware `dash.js` client is responsible for collecting the value of the CMCD parameters, including but not limited to current buffer length and measured throughput from different classes of the client implementation. After acquiring the desired values, the final CMCD query string gets generated by the CMCD-aware client. Subsequently, the CMCD query string is sent to the CDN/MEC servers with the HTTP GET request using HTTPLoader.js class.

However, our proposed end-to-end synchronization mechanism based on the concept of the Program Clock Reference ensures the buffer status (in seconds) is appended into the segment request and the reference clock samples are injected into the packet stream on the streaming client side. On the receiver side (i.e., at the MEC server), it performs the recovery of the clock samples with an adjustable virtual clock. Subsequently, it determines the time (in seconds) elapsed since the video segment request was sent. Accordingly, it calculates the updated client buffer status from the client sent buffer status and the actual elapsed time of sending the segment request to the MEC server. Thus, the mirrored buffer at the MEC

server provides an accurate estimate of the client playback buffer.

6.2.3 Adaptive Segment Prefetching Decision Engine

The adaptive segment prefetching decision engine for every per-user per-session thread starts with zero assumptions about the client status. The initial few segments of minimum bitrate level were served to the client. This will assure a lower initial startup delay for the video session at the client. The initial few segments for every session were either served directly from the source streaming server or from the MEC cache (where the content provider selectively caches the initial video segments for the popular video items). Note that our proposed approach does not consider the caching of the initial segments (especially for popular videos). Such a feature is out of the scope of our work. Subsequently, the MEC server infers the virtual buffer and predicts the throughput leveraging the HMM-based throughput predictor at regular intervals. The adaptive prefetching technique guarantees the video segment availability at the MEC server before the client requests them. In order to make an informed prefetching decision, we consider an adaptation heuristic that is already employed at the streaming client. The goal of employing the bitrate adaptation heuristic similar to the client is of two-fold. First, this will be able to predict the client's future requests accurately and could lead to a higher prefetching segment hit ratio. Secondly, such an assumption enables multiple content providers to flexibly manage the MEC server resources using their own policies (unlike in CDNs where the CDN operators manage the cache).

The proposed adaptive prefetching decision engine's main objective is to improve the end-users video QoE by prefetching the video segments closer to the client, leveraging both the predicted throughput and virtual client buffer occupancy. This simply translates to two key decisions that need to be made during a prefetching operation and are discussed below:

- **D1:** How many segments need to be prefetched ahead of the client's request progress during a video session?
- **D2:** What should be the video bitrate level of each of the segments to be prefetched?

On the one hand, prefetching a few segments ahead means a higher risk of falling back to the traditional client-server approach. This could degrade the video QoE significantly.

On the contrary, prefetching too many number segments could result in an injudicious utilization of the MEC server resources. The higher number of segments occupy more storage space at the MEC server and stored for a longer duration before either served to the client or removed due to a segment miss.

To carefully address the trade-off, our proposed adaptive prefetching decision engine at the MEC server stays ahead of the client's progress by maintaining a lead of fixed time duration. In practice, this time duration is a configurable parameter. It can be pre-configured by the content providers at the beginning of the video session, which is high enough for the MEC server to maintain its advance over the client's request progress. The adaptive prefetching decision engine makes an informed decision by utilizing the predicted throughput and virtual client buffer occupancy. Subsequently, the prefetching decision engine schedules the advance video segment request(s) to the source streaming server. The prefetched video segments were then stored at the MEC server storage. If the video segment request received from the DASH client is available at the Mec server, it is directly served to the client (as illustrated in Fig. 6.1). In case of a segment miss (i.e., either the video segment is not available at the MEC server) or the video bitrate level of the segment requested by the client does not match with the prefetched segment bitrate level, the client request is forwarded to the source video server. The video segment for the forwarded request is then retrieved from the source server and served directly to the client.

6.3 Performance Evaluation

In this section, we evaluate our proposed adaptive segment prefetching strategy by conducting controlled experiments on a trace-driven network testbed. In this section, we also delve in-depth into the implementation specifics of each component of the experimental testbed setup. We also describe the details of the DASH video dataset, link bandwidth traces, baseline comparison schemes and evaluation metrics used in the performance evaluation.

6.3.1 Experimental Testbed Setup

The network topology of the testbed that is used in our experiments is shown in Fig. 6.3. The testbed consists of an HTTP streaming server, MEC server, 802.11n Wi-Fi router (as eNodeB), and mobile client(s). Both streaming server and MEC server runs on an

Ubuntu 18.04.5 LTS machine with a 64-bit Intel Core i7 (6th Gen) 3.4 GHz CPU and 8 GB memory. We chose `Jetty` (release 9.4.11.v20180605) [38], a Java-based open-source HTTP server implementation, to build our video streaming server and MEC server[1]. We have implemented five custom modules to perform the MEC server functionality (as shown in Fig. 6.4) in the Jetty. The control flow in our implementation has been illustrated in detail in Fig. 6.4. A brief description of the five custom modules is summarized as follows.

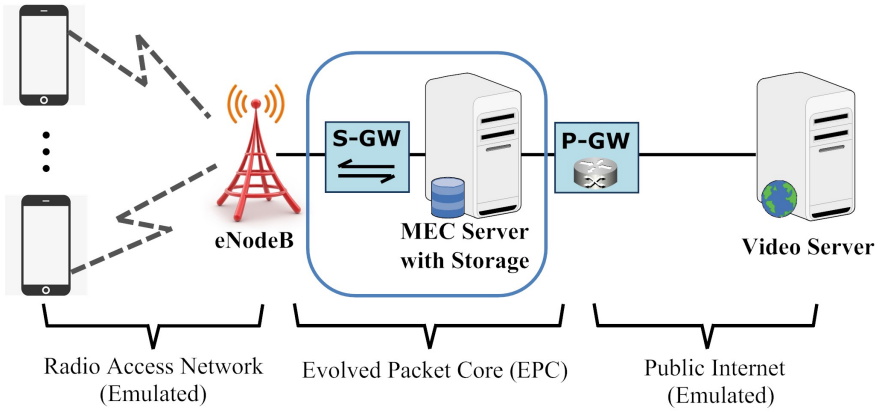


Figure 6.3: Overview of the trace-driven MEC emulation testbed network topology.

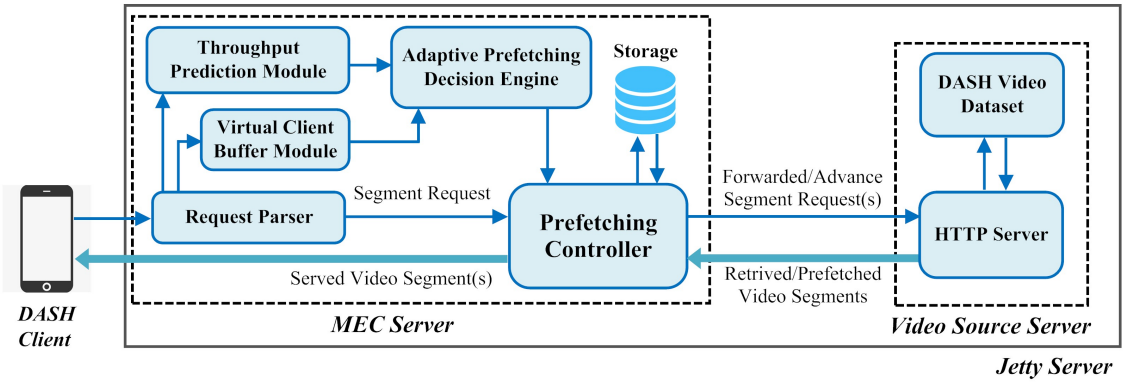


Figure 6.4: Overview of the MEC server workflow together with the custom implemented modules.

1. **Request Parser**: It parses the received client request and identifies the primary resource URI and associated resources array (including client buffer occupancy, client

[1] `https://github.com/hemakyarnagula/MECPrototypeCode/`

observed throughput, etc.). Subsequently, it pushes the primary resource URI to the prefetching controller module.

2. **Prefetching Controller**: Once the primary resource URI is received at the prefetching controller, it performs the following: 1) it verifies whether the segment request received from the DASH client is already prefetched and available in the MEC storage, 2) if the requested segment is available with the MEC server, it directly sends the response for the requested segment to the client from the MEC storage, and 3) in case of a segment miss, i.e., either the requested segment is not available, or the client requested video bitrate does not match with the prefetched segment bitrate level, the client request is simply forwarded to the video source server.

3. **Throughput Prediction Module**: The throughput prediction module is a trained HMM that leverages the throughput periodically disseminated through the MNO-owned RNIS module to predict the future throughput for the eNodeB-client link (as discussed in Section 6.2.1). We have implemented the HMM-based throughput predictor using Python. We have used the throughput measurements from the 4G/LTE network dataset [181] for the offline training of the HMM.

4. **Virtual Client Buffer Module**: The client playback buffer state is provided as an input to the virtual client buffer module (from the request parser). The virtual client buffer module then updates its local virtual client buffer status to maintain an accurate estimate of the client playback buffer (as discussed in Section 6.2.2). Thus, the virtual client buffer will felicitate to provide the client playback occupancy information to the prefetching decision engine without making an explicit request to the client.

5. **Adaptive Prefetching Decision Engine**: The adaptive prefetching decision engine uses a bitrate adaptation heuristic, similar to the one employed at the streaming client, to prefetch the video segments. It leverages both predicted throughput and client buffer status received from the throughput prediction module and virtual client buffer module, respectively, to make an informed prefetching decision (as discussed in Section 6.2.3). Subsequently, the prefetch decision engine informs its decision to the prefetching controller. The prefetching controller then schedules the advance video segment request(s) to the video streaming server. Subsequently, it stores the video

segments in the MEC storage once they are retrieved from the video source.

We have used the Samsung i9500 Galaxy S4 smartphone with an octa-core CPU and 2GB RAM (running Android 5.0.1) to conduct the experiments. For DASH streaming, we have used the `dash.js` framework [35] as the streaming client. The `dash.js` framework is an open-source JavaScript-based and potentially browser-independent DASH-compliant media player implementation. Our client implementation is based on the dash.js master branch (v1.2.0 release), as it was the stable version at the time of development. For our experimentation, the original rule-based bitrate adaptation logic of the adaptation module in the `dash.js` is replaced with custom adaptation schemes (refer Section 6.3.3).

DASH Video Dataset: To evaluate the performance of the proposed ASPF strategy, we have used the Tears of Steel (ToS), an SCI-FI movie sequence from the DASH dataset [153]. We have considered eight different representation/bitrate levels for the video sequence, consistent with the levels used by YouTube [155]. The total video length was about 735 seconds and consisted of 74 segments with a segment play duration of 10-seconds each. The segment duration (i.e., 10 seconds) is consistent with the range of chunk durations that are widely used in real-world commercial services [156]. The video resolution and encoding rate of the bitrate levels for the test video sequence are listed in Table 6.1.

Table 6.1: Summary of video resolutions and maximum encoding rate (in Mbps) for Tears of Steel video sequence.

RI	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
MER	0.50	0.81	1.5	2.4	3.0	4.0	6.0	10.0
VR	640x360	640x360	1280x720	1280x720	1920x1080	1920x1080	1920x1080	1920x1080

RI: Representation Index
MER: Maximum Encoding Rate (in Mbps)
VR: Video Resolution

Link Bandwidth Traces: Our key motive is to evaluate the performance of the proposed ASPF strategy using link conditions as observed in a radio network with mobility scenarios and a realistic backhaul network. In order to perform a realistic evaluation, we have used the throughput measurements from 4G/LTE network dataset [181] to emulate a radio network with vehicular mobility scenarios. The bandwidth and latency measurements in the dataset were collected in real 4G/LTE networks using different modes of transportation within the city of Ghent, Belgium. We randomly considered throughput traces collected using three modes of transportation, namely train, bus, and car, from the full dataset for

performing our experiments. These traces consist of continuous 1 second bandwidth and latency measurements recorded at the application-level while downloading a large file in an Android client. These traces not only vary in terms of the average throughput perceived but also in terms of their throughput profile. For example, the average throughput observed for the trace collected on a train around the city was 21.2 Mb/s $\pm$ 14.6 Mb/s. The significant distinction of the throughput profile across the traces mentioned above is primarily due to their measured location (i.e., bad coverage in general due to tunnels and large buildings) and selected route.

To the north of the evolved packet core (EPC) in Fig. 6.3, to emulate different public Internet conditions between the public Gateway (P-GW) and the video source server, we set the network parameters according to Sprint's monitored network service-level agreement performance in 2020 [182]. The network parameters between P-GW and the video server are set into the following scenarios:

1. Backhaul Link Capacity: We set the link capacity to 1 Gbps (adequate to represent an over-provisioned backhaul link). Note that we have varied the backhaul link load between 0% and 60%. In our experiments, the MEC server and the video source uses long-lived TCP connections and are not significantly affected by background traffic in the backhaul.

2. Latency and Packet Loss Ratio (PLR): For our experiments, we have considered the following latency (i.e., round-trip time) and PLR scenarios.

   (a) 500 ms and 0.05%: this represents the unconventional cases where the round-trip time is excessively high due to abnormal events such as BGP rerouting in the public Internet.

   (b) 300 ms and 0.05%: this represents the scenario where the video server is at a long-range location (e.g., North America to East Asia).

   (c) 200 ms and 0.04%: this represents the scenario where the video server is at a mid-range location (e.g., Europe to East Asia).

   (d) 100 ms and 0.03%: this represents the scenario where the video server is at a short-range location (e.g., Europe to North America).

These scenarios mentioned earlier will be referred to as 500ms, 300ms, 200ms, and 100ms scenarios, respectively, in the rest of the chapter. Moreover, we do not consider the case where the video server is available at a local source (i.e., within the same country) and the link has backhaul latency $\leq$ 10ms and PLR as 0.01%. We have also observed no significant improvements in QoE with our proposed ASPF approach in this scenario.

6.3.2 Baseline Comparison Schemes

In addition to our proposed ASPF scheme, we have considered two video segment delivery schemes from the literature to evaluate the overall performance. A brief summary of all the video segment delivery schemes used for comparison are as follows:

1. **Client-to-Server (C2S)**: This is the conventional DASH segment delivery scheme. In C2S, the video segments are streamed to the clients (i.e., to end-user devices) directly from the source server and no prefetching is performed. The MNO infrastructure neither provides dedicated storage for segments nor deploys additional intelligence in this scheme.

2. **Four Segments Ahead (FSA)**: We employed the prefetching scheme suggested by Krishnamoorthi et al. in [33], where the representative prefetching agent always aggressively prefetch up to four segments ahead of the currently requested segment by the streaming client. This work in [33] recommends prefetching either 1 or 4 segments in advance for HAS. We consider four video segments due to their relatively better performance over others. The four segments ahead scheme is implemented at the MEC server without the adaptive prefetching function. Thus, the prefetching agent located at the MEC server simply prefetches four segments of bitrate quality level similar to the client's currently requested segment.

3. **ASPF**: Our proposed adaptive segment prefetching scheme is deployed at the MEC server. This scheme implements all the functionalities described in Sections 6.2.

6.3.3 DASH Adaptation Algorithms

For performance comparison purpose, we implement five popular DASH bitrate adaptation algorithms from the literature. We make the bitrate adaptation algorithms implementation code[2] publicly available for reproducibility. The summary of all the DASH bitrate adaptation algorithms used for comparison is as follows.

1. **KM:** This algorithm proposed by Miller et al. in [79] uses an estimated network throughput along with current buffer occupancy (in playback seconds) to decide a suitable bitrate for the next segment. The KM algorithm maintains three buffer thresholds, namely B_{min}, B_{low}, and B_{high} (in seconds of playback time) to improve the robustness of the algorithm arise due to network measurement uncertainties. In our implementation, we have set the values of the three buffer thresholds B_{min}, B_{low}, and B_{high} to 10, 20, and 60 seconds, respectively.

2. **FSRA:** The fairness-aware smooth rate adaptation (FSRA) approach considers a dual-threshold based adaptation scheme combining the playback buffer occupancy and the probed bandwidth. We employed the probe-based bandwidth estimation scheme as suggested in [74]. The probe-based bandwidth estimation includes two schemes: a logarithmic law-based increase probing scheme (to avoid over-probing) and a conservative back-off based decrease probing scheme (to avoid congestion). In addition, it uses two operation thresholds to mitigate the effect of bandwidth variations and to obtain the best video bitrate for the future segment(s).

3. **ABMA+:** The adaptation and buffer management algorithm [40], called ABMA+, merely considers the predicted video playout rebuffering probability as a marker for adaptation. The ABMA+ uses a precomputed buffer map to avoid heavy on-line computations. We employed the precomputed buffer map suggested by Beben et al. [40] to determine the size of the playback buffer, which in turn, ensures a given rebuffering probability. In our implementation, we have assumed the rebuffering probability threshold equal to 10^{-4}, anti-oscillation factor equal to 0.1, and a buffer capacity of 60 seconds.

[2] `https://github.com/hemakyarnagula/DASHAlgorithmCode/`

4. **BOLA:** We have employed the function suggested in [37] for the Buffer Occupancy based Lyapunov Algorithm (BOLA). The BOLA for bitrate adaptation makes a control decision by merely utilizing the current buffer level (measured as the number of segments). Besides, the BOLA algorithm does not require explicit prediction of the available network bandwidth for making the control decisions. In our implementation, we have set the input weight parameter values for prioritizing playback utility (γ) and control parameter related to the maximum buffer size (V) to 0.5/segment play duration and 0.93, respectively.

5. **SARA:** The segment-aware rate adaptation (SARA) algorithm presented in [39] was implemented and the MPD files were enhanced by adding the segment size information. The SARA algorithm considers the impact of the segment size variation on the measured throughput and current playback buffer length along with the estimated network throughput to decide an appropriate bitrate level for the next segment. For our implementation, the SARA buffer thresholds were configured as $I = 2$, $B_\alpha = 5$, $B_\beta = 10$, and $B_{max} = 12$. These buffer thresholds used for buffer occupancy decide the size of the playback buffer reservoir (in seconds).

6.3.4 Evaluation Metrics

To evaluate the effectiveness of the proposed prefetching algorithm, we use the following performance metrics that are easy to determine either at the client or the MEC server.

1. **Prefetched Segment Hit Ratio (PSHR)**: It is defined as the percentage of segments that have been prefetched and served directly from the MEC server when the streaming client requests them.

2. **Prefetched Byte Hit Ratio (PBHR)**: Obviously, with more bytes retrieved and served directly at the MEC server, higher utilization of the backhaul link can be achieved. The PBHR metric is defined as the ratio between the number of bytes served directly from the MEC server storage over the total number of bytes prefetched from the source video server. Thus, it measures the effectiveness of the segment prefetching strategy in terms of the backhaul link capacity utilization.

3. **Average Segment Wait Time (ASWT)**: The average segment waiting time is defined as the average of the amount of waiting time spent by each segment in the MEC server storage before it is either served to the client or gets discarded (due to a segment miss). The objective of this metric is to minimize the average waiting time of the segments. Thus, our objective directly reflects judicious utilization of the resources (i.e., storage) at the MEC server.

4. **Objective QoE Metrics**: To evaluate the effectiveness of our proposed adaptive segment prefetching strategy in improving the end-user viewing experience, we consider the set of objective QoE metrics that are formally defined in Chapter 3.5.1. For completeness, we are enumerating the objective QoE metrics used for the evaluation of QoE below.

   1) Bitrate Adaptation Efficiency (BAE)
   2) Interruption Ratio (IR)
   3) Average Interruption Duration (AID)
   4) Bitrate Switching Amplitude Reward (BSAR)
   5) Video Continuity Index (VCI)
   6) Initial Startup Delay Reward (ISDR)

6.3.5 Results and Discussions

In this subsection, we evaluate our proposed ASPF approach from two different perspectives. First, we explore the impact of the ASPF decision engine's prefetch decisions while streaming with the SARA algorithm and different network link conditions. Second, we observe how the proposed ASPF approach fares against baseline schemes when the video gets streamed using popular DASH bitrate adaptation algorithms. We have also dive deeper into ASPF performance and demonstrate the improvements observed for the objective QoE metrics. Note that we have reported the mean value of the evaluation metrics obtained from multiple experiments with a 95% confidence interval (CI).

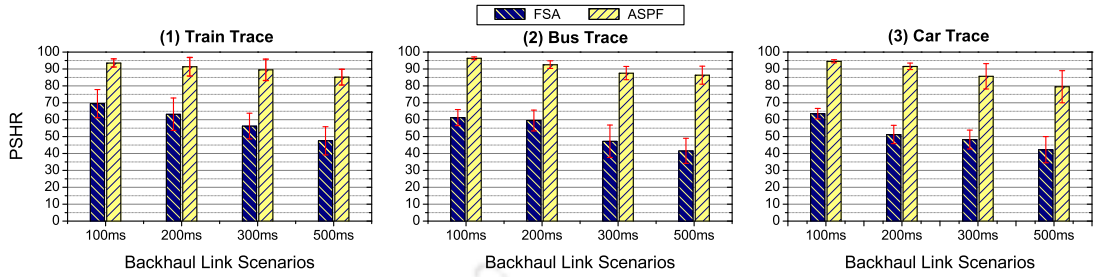
Single Client, Single Adaption Algorithm vs. Different Backhaul Link Conditions:

In the following subsections, we present the results of the experiments conducted using a single client to stream the video sequence (refer to Table 6.1) using the SARA algorithm. We have considered three different 4G/LET measurement traces, namely train, bus, and car, to emulate the real RAN environment together with the different backhaul link scenarios (refer to Section 6.3.1). The main reasons for using a single DASH adaptation algorithm, i.e., SARA are: i) to evaluate performance when the RAN link capacity is always sufficient (i.e., not the bottleneck) for the video since the RAN link bandwidth is sufficiently higher than the maximum available video bitrate; and ii) to demonstrate that even in such an ideal network environment at the RAN link, the conventional C2S scheme and the FSA prefetching scheme fare poorly, while our proposed ASPF scheme could able to improve end-users QoE compared to the other schemes. The performance results of the single client streaming video using the SARA algorithm are presented in Fig. 6.5 together with the QoE metrics performance statistics are summarized in Table 6.2.

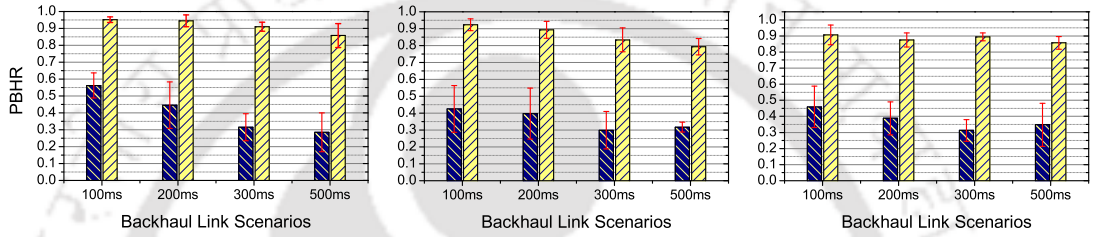
Note that the bitrate adaptation algorithm's ultimate goal is to download segments of a higher bitrate while avoiding unnecessary interruptions. We first look at the results under the 500ms scenario with all three RAN traces summarized in row 3-8 of Table 6.2. It can

Table 6.2: Performance statistics for the SARA algorithm under different backhaul link-LTE trace combinations.

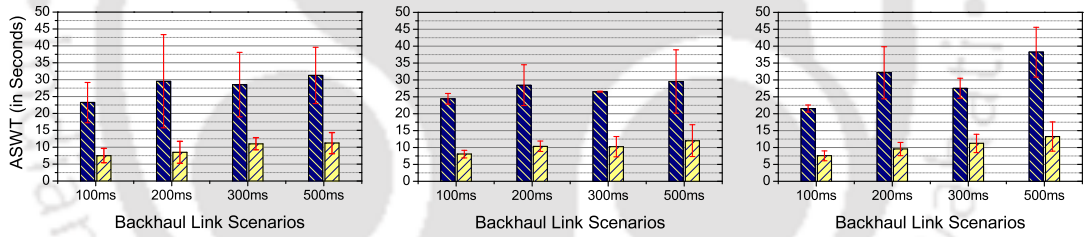
		Train Trace			Bus Trace			Car Trace		
		C2S	FSA	ASPF	C2S	FSA	ASPF	C2S	FSA	ASPF
500 ms and 0.05%	BAE	0.45440	0.63125	0.78652	0.47285	0.66548	0.75610	0.43124	0.62687	0.81475
	IR	0.04354	0.00324	0.00	0.05024	0.00524	0.00	0.01978	0.00275	0.00
	AID	2.01240	1.08457	0.00	1.97510	0.41058	0.00	2.24904	0.00645	0.00
	BSAR	1.67551	2.13321	2.58672	1.93932	2.16380	2.26634	1.77067	1.91101	2.18021
	VCI	0.99726	0.99852	1.00	0.99731	0.99944	1.00	0.99694	0.99999	1.00
	ISDR	0.02437	0.02794	0.04821	0.04054	0.04405	0.06398	0.03418	0.05418	0.01905
300 ms and 0.05%	BAE	0.48268	0.63706	0.79712	0.51524	0.67901	0.81248	0.48745	0.64104	0.82747
	IR	0.00875	0.00	0.00	0.00647	0.00354	0.00	0.00321	0.00	0.00
	AID	1.01647	0.00	0.00	0.87241	0.34872	0.00	1.0544	0.00	0.00
	BSAR	1.80048	2.38490	2.47383	1.87573	2.18779	2.58898	1.71184	1.81938	2.26606
	VCI	0.99861	1.00	1.00	0.99881	0.99952	1.00	0.99856	1.00	1.00
	ISDR	0.07293	0.07621	0.12605	0.01387	0.01735	0.07037	0.01123	0.01484	0.03538
200 ms and 0.04%	BAE	0.53721	0.64347	0.84512	0.53948	0.67584	0.86457	0.51245	0.67845	0.83544
	IR	0.00478	0.00	0.00	0.00215	0.00187	0.00	0.00547	0.00	0.00
	AID	0.82450	0.00	0.00	0.39875	0.03544	0.00	0.72570	0.00	0.00
	BSAR	2.08419	2.27424	2.57475	1.69856	2.18931	2.57203	1.97642	2.19501	2.43166
	VCI	0.99887	1.00	1.00	0.99945	0.99995	1.00	0.99901	1.00	1.00
	ISDR	0.04947	0.03576	0.06625	0.07487	0.06152	0.09119	0.02535	0.01201	0.01787
100 ms and 0.03%	BAE	0.54765	0.65281	0.87335	0.56124	0.65475	0.91584	0.52784	0.69741	0.87141
	IR	0.00984	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	AID	0.04580	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	BSAR	1.95886	2.36386	2.64971	1.89952	2.27424	2.67858	2.07104	2.38463	2.571835
	VCI	0.99993	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ISDR	0.10040	0.09686	0.11248	0.17757	0.17434	0.18862	0.11917	0.06871	0.08433



(a) Prefetched Segment Hit Ratio (PSHR)



(b) Prefetched Byte Hit Ratio (PBHR)



(c) Average Segment Wait Time (ASWT)

Figure 6.5: Performance statistics with single client and the SARA algorithm: (a) PSHR (b) PBHR and (c) ASWT.

be easily observed from Table 6.2 that the proposed ASPF scheme significantly outperforms both C2S and FSA schemes in terms of BAE, IR, AID, and BSAR. When compared with the FSA scheme, we have observed an improvement of 25%, 14%, and 30% for BAE with the train, bus, and car traces, respectively. Specifically, when compared with the C2S scheme, the ASPF scheme dramatically reduced its IR and AID. From Table 6.2, it is evident that the ASPF scheme could able to reduce the AID from around 2 seconds to 0 seconds (under the 500ms scenario) for all RAN traces. The reduction of the AID value in the 500ms scenario also resulted in increased VCI. These prove that with the ASPF scheme, we can

realize seamless DASH video delivery under such extremely challenging scenarios.

The C2S scheme is unable to perform well in terms of the QoE metrics (refer to Table 6.2). It could mainly be attributed to the high latency and packet loss in the backhaul link. The proposed ASPF scheme performance improvement over the FSA scheme is due to the ASPF scheme's adaptive segment prefetching capability, which is able to achieve a PSHR of 85%, 86%, and 79% for the train, bus, and car trace, respectively under the 500ms scenarios (shown in Fig. 6.5a). In contrast, the FSA scheme is able to correctly prefetch only 47%, 41%, and 42% of all segments with the train, bus, and car trace, respectively (as shown in Fig. 6.5a). The low PSHR values recorded by the FAS scheme are because merely prefetching four segments (with video bitrate similar to the previous segment) in advance is not enough to match the client's playback progress. Fig. 6.5a demonstrated a similar improvement in PSHR by the ASPF scheme under all the experimental scenarios considered (i.e., RAN trace-backhaul combinations) when video streamed using the SARA algorithm. The improvement in PBHR between the proposed ASPF scheme and the FSA scheme can be observed in Fig. 6.5b. The higher PBHR value observed for the ASPF scheme (as shown in Fig. 6.5b) is mostly attributed to its higher segment hit ratio under all scenarios. In contrast, the lower PBHR for the FSA scheme is mostly due to the prefetching of higher bitrate segments and evicting them from the MEC storage due to a subsequent segment miss. The ASWT observed per video playback session is shown in Fig. 6.5c. We observe that the ASPF scheme has comparatively lower ASWT compared to the FSA scheme. With the ASPF scheme deciding the number of segments to avoid excessive prefetching, the observed ASWT improvements also demonstrate that the ASPF scheme utilizes the MEC storage resources judiciously.

We next examine the QoE performance results under the 100ms scenario. It has been observed from Table 6.2 (see row 21-26) that our proposed ASPF scheme still outperforms both the baseline schemes, but the gap between their performance is marginal. Specifically, all three schemes have achieved zero IR and AID when the video gets streamed with the three RAM traces. It is also observed that the FSA scheme is eventually managed to achieve a BAE of around 0.65 for all three RAN trace scenarios. This is mostly because the client's buffer filled up quickly after a segment miss at the MEC server with abundant RAN capacity and lower backhaul access latency. In contrast, the ASPF scheme has achieved an average

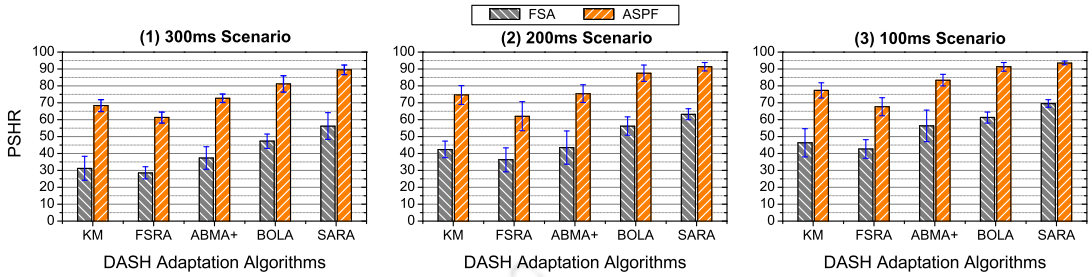
of 90% BAE for all three RAN trace scenarios. We have also observed no interruptions and higher BSAR with the ASPF scheme for all three trace scenarios.

Single Client, Different Backhaul Link Conditions vs. Five Different Adaption Algorithms:

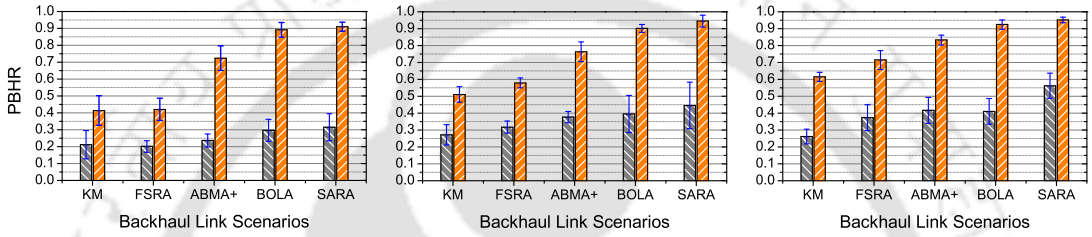
We now investigate the performance results of a client streaming video with five different DASH bitrate adaptation algorithms (refer Section 6.3.3), which are presented in Fig. 6.6 with statistics summarized in Table 6.3.

The first key observation is that with all the five DASH adaptation algorithms, the ASPF scheme is able to improve both PSHR and the video QoE during the video playback. Specifically, the ASPF scheme has achieved zero IR and AID compared to the other two baseline schemes when streamed using the SARA and BOLA. This because both SARA and BOLA managed to prefetch almost all the video segments (with appropriate video bitrate level) well in advance. Form Fig. 6.6a, we observe that the ASPF scheme with the SARA algorithm achieved a PBHR of 89.5%, 91.3% and 93.5% under 300ms, 200ms and 100ms scenarios, respectively. Similarly, the ASPF scheme with BOLA achieved a PBHR of 81.2%, 87.4% and 91.3% under 300ms, 200ms and 100ms scenarios, respectively. We had also observed a slight performance degradation for PSHR when video streamed using both KM and FSRA compared to the other adaptation algorithms. With the FSRA algorithm, the ASPF scheme is managed to accurately prefetch 61.2%, 62.1% and 67.6% of all segments on time under 300ms, 200ms and 100ms scenarios, respectively (as illustrated in Fig. 6.6a). This shows that the ASPF scheme, which is able to manage a performance improvement compared to the baseline approaches, could also result in poor viewing experience (i.e., degraded performance in QoE) due to the inaccurate throughput estimation by the bitrate adaptation algorithm employed at the client and frequent buffer fluctuations at the beginning of the video session.

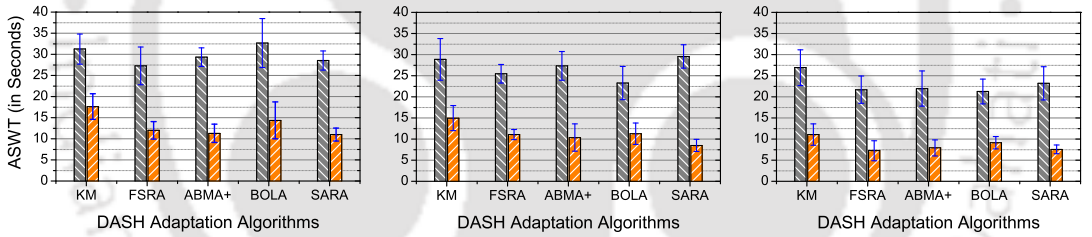
Unlike the proposed ASPF scheme, the FSA scheme has achieved a significantly lower performance, especially for ASWT, when the video gets streamed under 300ms, 200ms and 100ms scenarios (as shown in Fig. 6.6c). This is intuitive because, under the same backhaul link condition, the FSA scheme always prefetch a higher number of segments well in advance without considering the parameters affecting the segment adaptation decision(s) at the client. Thus, it stores them in the MEC storage for a longer duration before evicting them



(a) Prefetched Segment Hit Ratio (PSHR)



(b) Prefetched Byte Hit Ratio (PBHR)



(c) Average Segment Wait Time (ASWT)

Figure 6.6: Performance statistics with single client and different bitrate adaption algorithms: (a) PSHR (b) PBHR and (c) ASWT.

due to a segment miss. For example, under the 200ms scenario, the FSA scheme recorded an ASWT of 29.54 seconds (compared to only 8.47 seconds with the ASPF scheme) for SARA and 23.26 seconds (compared to only 11.24 seconds with the ASPF scheme) for BOLA, which is shown in Fig. 6.6c. This is mostly because the FAS scheme prefetch four segments (with video bitrate level similar to the last client requested bitrate level) in advance and only able to serve 56.2% and 63.2% of all segments prefetched to the client correctly for BOLA and SARA, respectively. This means that under the 200ms scenario, the FSA scheme buffers the prefetched video segments for a sufficiently long duration in the MEC storage

Table 6.3: Performance statistics for the five bitrate adaption algorithms under different backhaul link scenarios.

		300ms and 0.05%			200ms and 0.04%			100ms and 0.03%		
		C2S	**FSA**	**ASPF**	**C2S**	**FSA**	**ASPF**	**C2S**	**FSA**	**ASPF**
KM	BAE	0.431	0.512	0.612	0.479	0.536	0.631	0.513	0.593	0.681
	IR	0.136	0.113	0.013	0.125	0.100	0.024	0.085	0.101	0.124
	AID	8.325	2.739	1.264	5.360	3.102	0.746	4.936	1.356	0.365
	BSAR	1.003	1.259	1.322	1.178	1.209	1.357	1.271	1.167	1.357
	VCI	0.989	0.996	0.998	0.993	0.996	0.999	0.993	0.998	1.000
	ISDR	0.023	0.119	0.072	0.146	0.124	0.201	0.169	0.065	0.212
FSRA	BAE	0.247	0.336	0.580	0.277	0.438	0.613	0.351	0.482	0.632
	IR	0.041	0.026	0.003	0.050	0.035	0.004	0.033	0.028	0.003
	AID	7.125	2.046	0.031	3.649	1.942	0.064	4.297	1.617	0.007
	BSAR	1.149	1.181	1.495	1.258	1.396	1.627	1.291	1.490	1.552
	VCI	0.990	0.997	1.000	0.995	0.997	1.000	0.994	0.998	1.000
	ISDR	0.089	0.057	0.140	0.079	0.035	0.155	0.105	0.058	0.190
ABMA+	BAE	0.413	0.473	0.714	0.437	0.574	0.740	0.496	0.650	0.764
	IR	0.122	0.014	0.000	0.095	0.010	0.000	0.065	0.013	0.000
	AID	2.619	1.706	0.000	3.155	1.658	0.000	1.265	0.751	0.000
	BSAR	1.108	1.350	1.910	1.242	1.646	2.119	1.324	1.538	2.125
	VCI	0.996	0.998	1.000	0.996	0.998	1.000	0.998	0.999	1.000
	ISDR	0.081	0.124	0.059	0.078	0.080	0.086	0.190	0.134	0.084
BOLA	BAE	0.347	0.459	0.744	0.385	0.525	0.780	0.469	0.596	0.812
	IR	0.013	0.095	0.000	0.010	0.000	0.000	0.004	0.000	0.000
	AID	0.948	0.101	0.000	1.255	0.000	0.000	0.957	0.000	0.000
	BSAR	1.325	1.866	2.136	1.624	1.884	2.094	1.515	2.119	2.326
	VCI	0.999	0.999	1.000	0.998	1.000	1.000	0.999	1.000	1.000
	ISDR	0.068	0.019	0.079	0.064	0.110	0.161	0.068	0.118	0.078
SARA	BAE	0.483	0.637	0.797	0.537	0.643	0.845	0.548	0.653	0.873
	IR	0.009	0.000	0.000	0.005	0.000	0.000	0.010	0.000	0.000
	AID	1.016	0.000	0.000	0.825	0.000	0.000	0.046	0.000	0.000
	BSAR	1.800	2.385	2.474	2.084	2.274	2.575	1.959	2.364	2.650
	VCI	0.999	1.000	1.000	0.999	1.000	1.000	1.000	1.000	1.000
	ISDR	0.073	0.076	0.126	0.049	0.036	0.066	0.100	0.097	0.112

before removing them due to segment miss at the MEC server. Thus, buffering the segments for a longer duration by FSA scheme results in higher ASWT (as shown in Fig. 6.6c) and makes injudicious utilization of the MEC storage resources. In contrast, the proposed ASPF scheme dynamically adjusts the number of segments for prefetching, leveraging the client buffer status (using an emulated virtual client buffer) while maintaining a sufficient time gap ahead of the client request progress.

We observe that for all DASH bitrate adaptation algorithms under all scenarios, both FSA and ASPF schemes able to improve BAE, BSAR, and VCI over the C2S scheme, as

evident from Table 6.3. This is because, in the C2S scheme, the video segment always needs to be retrieved from the video source and cannot be prefetched. Thus, they incur the additional access latency of the backhaul link for every segment. However, it still can be observed that as the backhaul condition gets better (i.e., from 300ms to 100ms), the overall performance with the C2S scheme for all the adaptation algorithms gets improved (refer to Table 6.3). From Table 6.3, it has been observed that all the three schemes, including the proposed ASPF scheme, achieved similar performance for ISDR under all the experimental scenarios considered. The similar performance for ISDR is attributed to the lower or negligible improvement for the initial startup delay. Typically, the initial startup delay refers to the time duration required for a client to download and parse the MPD manifest file and fetch the first few segments. This is because the first few video segments always need to be retrieved from the source video server with all three schemes to build a sufficient buffer reservoir and can not be prefetched. Note that in our proposed ASPF scheme, we do not assume the pre-caching of the first few video segments at the MEC server. However, pre-caching a subset of the first few segments with a lower bitrate for popular videos could further reduce the initial start delay. The lower initial startup delay, i.e., starting video playback as soon as the first few segments are delivered to a client (which is the case in many DASH clients such as `dash.js`), will also lead to better user engagement [8].

Meanwhile, under all the three backhaul scenarios (i.e., 300ms, 200ms, and 100ms), both the ASPF and FSA schemes were able to support seamless playback for both SARA and BOLA, as reported in Table 6.3. This is because both SARA and BOLA maintain sufficient segments in the client buffer to support seamless playback and avoids interruptions arise due to RAN resource competition or fluctuations. It is worth noting that the proposed ASPF scheme can accurately prefetch most of the video segments to the MEC server from the remote video source to gain its advance over the client's request progress, as shown in 6.6a. In addition, it is able to record a higher BAE for all five algorithms. Nevertheless, clients employed with adaptation algorithms that maintain a lower buffer reservoir and choose the video bitrate just to saturate the available RAN capacity, such as KM and FSRA, still experienced smaller duration interruptions (as shown in Table 6.3) mostly due to RAN resource fluctuations.

Table 6.4 summarized the observed improvement for PSHR, BSAR, and BAE with the proposed ASPF scheme when Tears of Steel video sequence streamed via train trace measurements. The results in Table 6.4 demonstrate that our proposed ASPF scheme with the SARA algorithm is able to increase PSHR by 35%, BAE by 34%, and BSAR by 12% when compared under 100ms backhaul link scenario. In addition, our proposed ASPF scheme with the BOLA algorithm under 100ms backhaul link scenario able to improve BAE by 36%, increase PSHR by 49%, and BSAR by 10%. We also observe similar improvements for all the metrics, including influencing QoE metrics in all the experimental scenarios considered.

Table 6.4: Performance improvements (in percentage) for PSHR, BSAR, and BAE across all the backhaul link scenarios and with train trace.

Backhaul Scenarios	**300ms and 0.05%**			**200ms and 0.04%**			**100ms and 0.03%**		
Metrics	**PSHR**	**BSAR**	**BAE**	**PSHR**	**BSAR**	**BAE**	**PSHR**	**BSAR**	**BAE**
SARA	59.22	3.73	25.12	44.46	13.21	31.34	34.56	12.09	33.78
BOLA	71.92	14.43	62.10	55.48	11.17	48.53	48.99	9.73	36.22
ABMA+	94.52	41.46	50.73	73.06	28.79	28.85	47.96	38.23	17.52
FSRA	114.56	26.55	72.34	71.05	16.56	39.71	58.67	4.1	31.09
KM	118.68	5.00	19.51	76.16	12.28	17.67	66.87	16.28	14.79

QoE Score Prediction: To demonstrate the overall QoE improvements for the proposed ASPF scheme with the five different bitrate adaptation algorithms, we have considered the parametric MLR model (as discussed in Chapter 4) to predict the QoE score. To this end, we have provided the measured objective QoE metrics values as input to the parametric MLR model. The parametric MLR model, in turn, predicts an appropriate QoE score similar to the end-user perceived QoE. Table 6.5 summarizes the QoE score predicted for the five different bitrate adaptation algorithms with the three different segment prefetching schemes (discussed in Section 6.3.2). In Table 6.5, the values represent the mean of the QoE score obtained from multiple instances with the confidence interval in brackets. The predicted score demonstrates that all the five bitrate adaptation algorithms with the APSF scheme have outperformed the other two baseline comparison schemes (i.e., C2S and FSA schemes) across all the experimental scenarios as reported in Table 6.5. We have also observed that SARA, followed by BOLA, outperforms all the other adaptation algorithms in terms of the QoE score across all the three backhaul link scenarios (as reported in Table 6.5).

Table 6.5: Predicted QoE score (with confidence interval) by the parametric MLR model for the five bitrate adaptation algorithms.

		C2S	FSA	ASPF
300 ms and 0.05%	SARA	3.4286 (0.25560)	4.0357 (0.29360)	4.2857 (0.19799)
	BOLA	2.8519 (0.17202)	3.2963 (0.20433)	4.0741 (0.25467)
	ABMA+	2.7037 (0.20433)	3.2593 (0.22419)	3.8148 (0.27752)
	FSRA	2.6296 (0.18562)	2.9629 (0.19592)	3.3704 (0.23737)
	KM	2.5185 (0.19206)	2.9259 (0.17895)	3.2593 (0.16845)
200 ms and 0.04%	SARA	3.4643 (0.23604)	3.9643 (0.25666)	4.4286 (0.18666)
	BOLA	3.0370 (0.12733)	3.5185 (0.19206)	4.1481 (0.27086)
	ABMA+	2.8148 (0.21020)	3.3704 (0.18562)	3.9259 (0.25467)
	FSRA	2.7037 (0.17552)	3.0370 (0.19520)	3.4815 (0.19206)
	KM	2.7037 (0.20433)	2.9630 (0.12733)	3.2963 (0.17552)
100 ms and 0.03%	SARA	3.5714 (0.25560)	4.1071 (0.25382)	4.5714 (0.18666)
	BOLA	3.1481 (0.13655)	3.8148 (0.21020)	4.2593 (0.28826)
	ABMA+	2.9630 (0.19520)	3.4074 (0.18887)	4.0370 (0.24493)
	FSRA	2.8148 (0.14931)	3.1481 (0.17202)	3.5185 (0.19206)
	KM	2.9259 (0.20728)	3.0370 (0.19520)	3.2963 (0.17552)

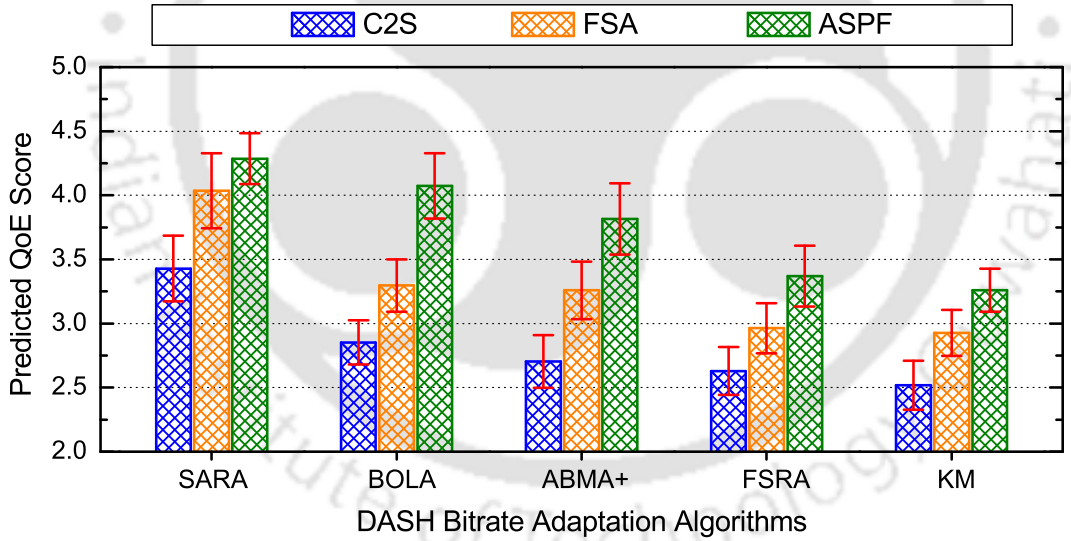


Figure 6.7: Illustration of QoE score with confidence interval predicted by the parametric MLR model for five different DASH adaptation algorithms with three video segment delivery schemes under 300 ms Latency and 0.05% Packet Loss Ratio scenario.

Furthermore, we have plotted the QoE score predicted with parametric MLR model across different Latency and Packet Loss Ratio scenarios (as reported in Table 6.5) to visually observe if the confidence intervals of the different video segment delivery schemes overlap with each other. The graphs illustrating QoE score with confidence interval across

five different DASH adaptation algorithms with three video segment delivery schemes under 300 ms Latency and 0.05% PLR, 200 ms Latency and 0.04% PLR, and 100 ms Latency and 0.03% PLR scenario are shown in Fig. 6.7, Fig. 6.8 and Fig. 6.9, respectively.

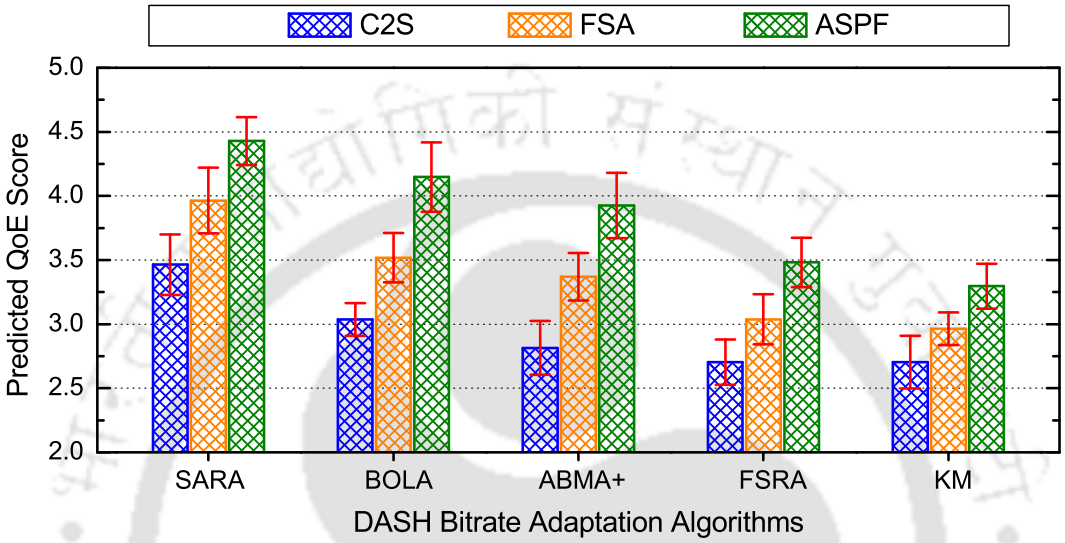


Figure 6.8: Illustration of QoE score with confidence interval predicted by the parametric MLR model for five different DASH adaptation algorithms with three video segment delivery schemes under 200 ms Latency and 0.04% Packet Loss Ratio scenario.

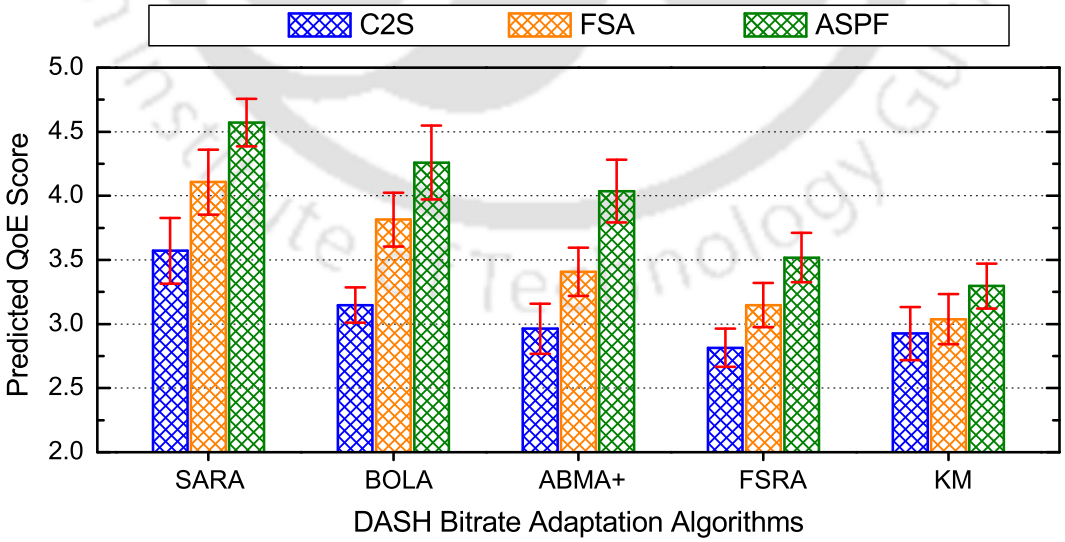


Figure 6.9: Illustration of QoE score with confidence interval predicted by the parametric MLR model for five different DASH adaptation algorithms with three video segment delivery schemes under 100 ms Latency and 0.03% Packet Loss Ratio scenario.

6.4 Summary

In this chapter, we proposed an adaptive segment prefetching strategy leveraging the MEC functionality with the objective of improving the overall QoE for DASH in mobile clients. The proposed scheme dynamically determines the number of video segments and their corresponding video bitrate level for prefetching ahead of the client's request progress. The prefetched segments are then served directly from the MEC server upon receiving the actual client request. In a nutshell, our proposed ASPF scheme works as an enhanced proxy implementing the opportune prefetching of video segments on a per-user per-session basis at the MEC server. To demonstrate the efficacy of the proposed ASPF scheme, we provide a practical implementation of a network trace-driven MEC-based streaming prototype with support for the adaptive segment prefetching mechanism. We have also implemented five popular DASH bitrate adaptation algorithms in the `dash.js` for evaluating the performance of the proposed ASPF scheme. The findings from our experimental results demonstrate that the ASPF scheme significantly outperforms the baseline schemes (i.e., traditional C2S scheme and FSA scheme) both in terms of segment hit ratio and overall QoE improvement across all the experimental scenarios. The experimental results have shown that the proposed ASPF strategy improved PSHR and BAE by 60% and 25%, respectively, when streamed with the SARA algorithm under the 300ms backhaul link scenario. We have also observed similar improvements for both PSHR and BAE leading to reduced segment access latency and increased QoE in all other scenarios. In addition, we demonstrated the efficiency of the proposed ASPF strategy through the predicted QoE score using the parametric MLR model.

CHAPTER

7

Conclusions and Future Directions

In this thesis, several contributions have been made to efficiently deliver DASH videos on mobile devices and improve the end-user QoE. In this section, we summarize the research contributions made in this thesis and discuss their impact. Next, we outline several research directions, including both intimidate extensions to the works carried in this thesis and currently arising in the field of multimedia delivery, which we believe will be relevant in future years.

7.1 Conclusions

First, we investigated how the bitrate adaptation algorithms in DASH influence the end-user viewing experience, particularly operating under highly fluctuating bandwidth conditions such as mobile networks. To this end, we have performed video QoE assessment (both subjective and objective assessments) of several popular DASH bitrate adaptation algorithms in mobile devices using a quality assessment experimental framework. We also presented and formally defined a set of application-layer objective QoE metrics for the objective quality assessment with an end goal to capture the severity of the metrics on the end-user viewing experience. We have also proposed a linear parametric QoE model to compute a cumulative QoE score which can be used for a fair comparison of the bitrate adaptation algorithms. Second, we have proposed a parametric QoE-estimation model based on multinomial logistic regression (MLR) for video quality estimation of DASH streams. The

proposed MLR model first gets trained using the video quality assessment results obtained from both subjective and objective QoE assessments. The trained QoE-estimation model then merely takes a set of five objective QoE metrics values to predict the overall QoE score as perceived by the end-user. We validated the quality-prediction accuracy of our trained QoE-estimation model using both hold-out and k-fold cross-validation techniques. Third, we mitigated the request explosion problem and addressed the over-push problem to improve the QoE of DASH. To this aim, we proposed an adaptive segment-aware K-push algorithm (termed ASK-Push) to determine the appropriate adaptation push pair (κ and $\hat{b}_\kappa$) for DASH/2. The proposed adaptive segment-aware K-push algorithm aims to improve QoE while addressing both the request explosion problem leveraging the server-push feature and the over-push problem in mobile clients. The proposed ASK-Push algorithm utilizes segment size information to estimate the time required to download the next κ segments in a push cycle. Subsequently, it takes network throughput estimated using weighted harmonic mean strategy, segment size information, and current playback buffer occupancy into account for deciding the appropriate video bitrate level ($\hat{b}_\kappa$) for the next push cycle in DASH/2. In addition, the proposed algorithm estimates the time required to wait before initiating a push cycle request to avoid superfluous playback buffer filling and thereby addresses the over-push problem in mobile devices. We provided a practical implementation of the DASH/2 streaming prototype with support for the adaptive-push mechanism and low-overhead client implementation for the ASK-Push algorithm. We performed an extensive systematic objective QoE assessment of the proposed ASK-Push algorithm and compared it with the existing DASH/2 solutions. Finally, we proposed an adaptive segment prefetching (termed ASPF) strategy with an aim to improve the end-user QoE for DASH in MEC networks. The proposed ASPF strategy dynamically determines the number of segments and their corresponding video bitrate leveraging the last-mile radio link throughput prediction and virtual client buffer. Subsequently, it prefetches the appropriate video segments at the MEC server while maintaining a sufficient gap ahead of the client's actual segment request progress. Once the client requests the video segments that are already prefetched, they get served directly from the MEC server. Thus, reducing the segment access latency and improving the overall QoE for the DASH video session. In a nutshell, our proposed ASPF scheme works as an enhanced proxy implementing the opportune prefetching of video segments on a

per-user per-session basis at the MEC server. We also provided a practical implementation of the MEC-based streaming prototype with support for the adaptive segment prefetching mechanism.

From the subjective QoE assessment, we found that the bitrate adaptation algorithms that consider a combination of estimated throughput, current buffer occupancy, and other parameters could able to provide better QoE to the end-users. Our objective QoE assessment results demonstrated that bitrate adaptation algorithms that follow an aggressive approach in selecting the segment bitrate able to achieve higher bitrate adaptation efficiency (i.e., serve segments of higher bitrate levels). On the other hand, the results also demonstrated that bitrate adaptation algorithms that follow a conservative approach in selecting the video bitrate for segments tend to achieve a lower interruption ratio (i.e., fewer interruptions for the entire video session) and a higher bitrate switching amplitude reward (i.e., smoother rendering of the video with fewer bitrate switching events). We observed from the video QoE assessment results that none of the bitrate adaptation algorithms performs well with respect to all the objective QoE metrics under any scenario. Thus, it is evident that there exists a mutual dependence among the objective QoE metrics. From the video QoE assessment of bitrate adaptation algorithms in mobile devices, we can conclude that, due to the trade-offs involved among the objective QoE metrics considered, no bitrate adaptation algorithm can lead to better QoE merely by selecting an appropriate bitrate for the next video segments. Besides, we have evaluated the bitrate adaptation algorithms performance based on the cumulative QoE score as computed by the proposed linear parametric QoE scoring model. We found that the bitrate adaptation algorithms that follow a conservative approach in selecting the video bitrate outperform in terms of the cumulative QoE score compared to the others. This is mostly due to the informed bitrate decision taken with the aim of minimizing the interruptions and bitrate switches while increasing the video bitrate efficiency by the bitrate adaptation algorithms. Thus, the bitrate adaptation algorithms with an objective to assure a satisfying QoE should try to improve all the QoE metrics rather than focusing on a subset of metrics.

For the proposed parametric MLR model, we observed a Pearson's linear correlation coefficient of 0.9235 and a root mean square error of 0.416 during the quality-prediction accuracy validation with the hold-out test dataset. The quality-prediction accuracy validation

results also demonstrated that the proposed parametric MLR model achieved an 83% prediction accuracy with the validation data. Moreover, we have observed an average of 81% prediction accuracy for the parametric MLR model with the k-fold cross-validation technique. Next, we assessed the impact of the training set size for the proposed parametric MLR model. We also observed that the proposed parametric MLR model could able to achieve a Pearson's linear correlation coefficient of 0.83056 even when trained with a training set consisting of 90 samples which is significantly higher than the minimum performance benchmark (i.e., "PLCC $\geq$ 0.70"). The validation results demonstrated the efficacy and applicability of the proposed MLR model under all the scenarios. Moreover, a comparison has been performed with three other well-known models in the literature, which confirmed that the proposed parametric MLR model could predict QoE with a reasonably low computational complexity, suggesting its suitability for online use. In addition, a scalability study has confirmed that the proposed MLR model can efficiently predict the overall QoE score for DASH videos of longer duration (i.e., with a play duration larger than 60 minutes).

We evaluated the efficiency of our proposed ASK-Push algorithm using a practical implementation that considers a set of throughput variability traces, different video datasets, and a set of influencing objective QoE metrics. The findings from our extensive evaluation demonstrated that the ASK-Push algorithm significantly outperforms existing schemes (i.e., adaptive-push scheme [26] and DASH2M scheme [27]) in terms of overall QoE improvement. The experimental results have demonstrated that the ASK-Push algorithm achieved an average of 6%, 67% and 19% improvement for bitrate adaptation efficiency, interruption ratio, and bitrate switching amplitude reward, respectively, compared to other approaches across all the considered experimental scenarios. Moreover, the objective QoE metrics measurements have confirmed that the proposed approach achieved playback smoothness while maintaining a significantly lower interruption ratio and a higher video continuity index. We also observed that the ASK-Push algorithm is able to achieve an average of 13% and 8% improvement in terms of the QoE score predicted using the parametric MLR model when the videos get streamed with train and car trace, respectively.

We evaluated and analyzed the efficiency of our proposed ASPF strategy using a set of throughput variability traces and backhaul link scenarios, different DASH adaptation algorithms, and a set of objective QoE metrics. The findings from our extensive evaluation

demonstrated that the ASPF strategy significantly outperforms the baseline schemes in terms of overall QoE improvement when compared in the presence of realistic network cross-traffic. We observed that the SARA algorithm with the proposed ASPF strategy improves bitrate adaptation efficiency by 25%, 31% and 34% when streamed under 300ms, 200ms and 100ms backhaul link scenarios, respectively. For the BOLA algorithm with the proposed ASPF strategy, we observed an improvement of 62%, 49%, 36% in terms of bitrate adaptation efficiency when streamed under 300ms, 200ms and 100ms backhaul link scenarios, respectively. The experimental results also demonstrated that the proposed ASPF strategy is able to achieve a significantly higher segment hit ratio at the MEC server for all the bitrate adaptation algorithms considered. We observed an average improvement of 59% for prefetched segment hit ratio when the SARA algorithm is streamed under the 300ms backhaul link scenario. Similarly, we observed an average improvement of 72% for prefetched segment hit ratio when the BOLA algorithm is streamed under the 300ms backhaul link scenario. We have observed similar improvements in terms of the prefetched segment hit ratio for all the bitrate adaptation algorithms and backhaul link scenario combinations. The higher prefetched segment hit ratio by the ASPF strategy also demonstrated a better utilization of the MEC-video server backhaul link capacity as compared to the existing baseline schemes. In addition, we have predicted the overall QoE score for the bitrate adaptation algorithms across all the experimental scenarios using the parametric MLR model. We observed that all the bitrate adaptation algorithms, when streamed with the proposed ASPF strategy, achieved a significantly higher QoE score compared to the baseline strategies across all the experimental scenarios.

7.2 Future Research Directions

As reported and analyzed throughout this thesis, several control actions have been proposed to improve the end-user QoE for DASH. Nevertheless, the works presented in this thesis can be extended in various directions and leaves ample scope to advance the work. We present a few immediate extensions to the contributions in this thesis below. In addition, several new challenges are emerging in the field of multimedia delivery, which will need to be addressed by the multimedia research community in the near future. In what follows, we have also identified the newly arising opportunities and the main challenges associated with them.

7.2.1 Immediate Extensions of Thesis Work

In this subsection, we list a few immediate extensions of the research works reported in the contributing chapters of this thesis.

- In our work, we assumed video sequences without advertisements to obtain participant's overall viewing experience during the subjective QoE assessment. The advertisements during a streaming session offer the service providers the opportunity to accumulate sufficient playback buffer to avoid the buffer depletions in the near future and improve viewer engagement. Nevertheless, based on various factors, including the type of advertisements, frequency and duration of advertisements per viewing session, and viewers' interest could lead to annoyance for the viewer. In the future, the impact of advertisements that include both non-skippable and skippable advertisements can be considered for capturing the participant's overall viewing experience during the subjective QoE assessment.

- The proposed parametric QoE-estimation model does not take the effect of advertisements on both the initial startup delay reward metric and participant's overall viewing experience during the subjective experiments. The future work could focus on the impact of initial advertisements on ISDR and BAE metrics and the effect of advertisements on participant's overall viewing experience during the video session to predict the QoE score.

- In our work, we considered pre-configured buffer thresholds for the buffer model of the ASK-Push algorithm, and these values remain constant for the entire video session. In future work, a dynamic buffer threshold management technique can be incorporated to explicitly tune the performance of the ASK-Push algorithm during multi-player interactions depending on the underlying network bandwidth variation. In addition, the future works could focus on evaluating the ASK-Push algorithm performance during multi-player interactions and address the multi-player QoE fairness issues with the change in the number of users/viewers.

- In our work, we assumed client devices associated with a single RAN base station (i.e., eNodeB) and with moderate-mobility for the adaptive segment prefetching.

Nevertheless, video segment delivery utilizing the adaptive segment prefetching strategy for clients with high-mobility will bring unique challenges. Moreover, the frequent hand-offs between the mobile base stations could significantly degrade the end-user QoE. The work can be extended by considering the adaptive segment prefetching together with the cooperative edge prefetching through intelligent adaptive learning and decision-making strategies in order to ensure satisfactory QoE even in challenging situations such as high-mobility.

- The contributions in this thesis mainly deal with video streaming for mobile devices. We have considered Full High Definition (FHD) resolution of 1920 x 1080 pixels as the highest bitrate for our experiments due to the smaller screen sizes of the mobile devices. Of late, the DASH-based Ultra High Definition (UHD) video streaming service is evolving rapidly. In the meantime, the resolution standards have also improved rapidly over the past few years: from Full HD to next-level Ultra HD (3840 x 2160 pixels), which was then replaced by 4K (4096 x 2160 pixels), to all-new 8K (7680 x 4320 pixels) with an aspect ratio of 16:9, which is already waiting in the wings, ready to take over. Moreover, Ultra High Definition (UHD) video requires more than four times the bandwidth used to stream the Full High Definition (FHD) video. The future work could focus on applicability and performance evaluation of the contributions in this thesis for streaming videos with higher resolutions (including UHD or 4K) to desktop/laptop systems with larger screens over the link equipped with the current wireless standards.

7.2.2 Immersive Video Streaming

In recent times, specialized head-mounted display (HMD) devices such as the Oculus Rift and HTC Vive hit the market and quickly becoming accessible to a large public. With the recent developments in Virtual Reality (VR) streaming technology, several new challenges have been arisen for supporting these applications for entertainment, education, and commerce. Although these technologies are still in their infancy, it is expected that their demand will grow consistently in the near future and they have huge potential to shape the next experience. In VR streaming, the viewer is completely immersed in a virtual environment. The viewer can freely decide the preferred viewing position of the 360° immersive video,

also called viewport. Unfortunately, the VR streaming applications are often affected due to the high bandwidth requirements of the 360° video and could only render lower quality viewports to the end-users. Therefore, this could severely affect the end-user QoE. The end-users can also experience symptoms of motion sickness that include but are not limited to eye fatigue, disorientation, and nausea.

To address the issues mentioned above, several viewport-dependent VR streaming solutions have often been proposed with an objective to reduce the bandwidth requirement to stream the VR videos. In viewport-dependent VR streaming, the portion of the 360° immersive video actually watched by the viewer (also called known as Field-of-View) is only streamed at the highest quality. On the contrary, the video outside the Field-of-View can be streamed at a lower quality or not streamed. Tile-based viewport-dependent VR streaming [183] is extremely interesting in the HTTP adaptive streaming domain. The tile-based viewport-dependent VR streaming can be obtained by spatially tiling the 360° video. To support the tile-based viewport-dependent VR streaming, MPEG-DASH has recently standardized a new specification, called spatial representation description [184]. In the context of tile-based viewport-dependent VR streaming, we list a few important challenges that need to be addressed to ensure a satisfactory end-user QoE.

- First, to prioritize the delivery of video tiles in the viewer Field-of-View as opposed to tiles outside the Field-of-View. Clearly, this translates to delivering the video tiles in viewer Field-of-View with higher quality as quickly as possible than the others. The new HTTP/2 protocol, with its new features such as server push, stream prioritization and multiplexing, can be leveraged to develop a possible solution to prioritize the delivery of the tiles in the Field-of-View.

- Second, to predict the Field-of-View which a viewer is going to watch in the near future. Since the viewer is immersed in a virtual environment, they can reach regions of the 360° video at lower qualities by moving their heard. Thus, it is crucial to predict these changes beforehand in order to provide a seamless transition when the viewer moves from one region to another.

- Finally, to investigate the effect of watching regions of the 360° video at different qualities on end-user QoE. Despite the benefits of Field-of-View prediction algorithms

and adaptation algorithms for selecting appropriate tile quality, the viewer can still be presented with tiles at different quality levels. Thus, the impact of this behavior on end-user QoE needs to be investigated to provide insights for designing tile-based bitrate adaptation heuristics with an objective to assure satisfying QoE to end-users.

7.2.3 Multimedia Analytics

In the last decade, we have witnessed widespread use of multimedia such as video, audio, and image, and the availability of inimitable and valuable multimedia sources. This has resulted in a big data revolution in multimedia management systems. This rich source of multimedia information, often considered "big data" (due to its high volume and variety), are usually unstructured and noisy. Therefore, it is not feasible to efficiently and effectively analyze the huge amount of multimedia data using conventional big data analytics methods in order to gain insight and knowledge.

Multimedia analytics [185] addresses the issue of manipulating, managing, mining, understanding, and visualizing different types of multimedia data in effective and efficient ways to solve real-world challenges. At present, multimedia analytics techniques are used to manipulate multimedia data sensibly and cost-effectively. Current and potential multimedia analytics applications [186] include but are not limited to surveillance video framework, multimedia summarization, social networks, and disaster management systems. In the light of the wide adaptation, we list several research directions that we believe must be addressed in order to make progress towards scalable multimedia analytics.

- With the advances in camera and smartphone technology, we have witnessed not only the volume of multimedia items but also the size per item in virtually all application domains. To serve the purpose, an efficient Multimedia Database Management System (MMDBMS) should be designed to support the different workloads that arise along and facilitate multimedia analytics for increasingly large-scale multimedia data. Besides, the MMDBMS needs to meet the following requirements [187]: traditional DBMS capabilities, huge capacity storage management, multimedia query support, and information retrieval capabilities.

- Multimedia visualization tools are a crucial requirement of multimedia analytics to improve the user's interaction. The visualization tools can assist in the three

major types of analytics: descriptive, predictive, and prescriptive [188]. Nevertheless, the computational complexity of the analytical operations degenerates the real-time performance [189]. To this end, efficient visualization tools need to be developed that adapt to both the amounts and types of data while improving both the user interactivity and performance.

7.2.4 Adaptive Bitrate Streaming over QUIC

Of late, Google proposed a new transport layer protocol called the Quick UDP Internet Connection (QUIC) protocol [190]. The QUIC, based on UDP, can be implemented in the user space rather than the kernel. Thus, it can be more easily deployed and updated as compared to TCP. The QUIC provides several interesting improvements as compared to TCP. First, the 0-RTT connection establishment reduces the latency if the client and server have already communicated in the past. Second, true multiplexing of HTTP/2 streams at the transport level as compared to TCP. The standard TCP still introduces head-of-line blocking when the packets of a particular stream are lost. Third, QUIC provides the possibility to easily deploy new congestion control algorithms to guarantee reliability. The significant innovations introduced by the QUIC can be useful in adaptive bitrate streaming and is especially beneficial when network congestion, packet loss, and RTT are high. Nonetheless, a complete analysis of adaptive bitrate streaming over QUIC has only been marginally investigated [191] in the literature. In the light of the above, we list a few challenges that need to be addressed.

- To formally evaluate the impact of adaptive bitrate streaming over QUIC on the end-user viewing experience.
- To investigate the network scenarios in which the adaptive bitrate streaming over QUIC outperform the standard TCP.
- To explore possibilities to develop a congestion control algorithm for QUIC, which is specifically tailored towards improving the end-user QoE during the adaptive streaming session.

Bibliography

[1] C. Timmerer, *MPEG-CMAF: Threat or Opportunity?*, Bitmovin, August 2016, Accessed on Feb. 10, 2022. [Online]. Available: https://bitmovin.com/what-is-cmaf-threat-opportunity/

[2] C. Timmerer, *MPEG-DASH: Overview, State-of-the-Art, and Future Roadmap.*, October 2016, Accessed on Feb. 10, 2022. [Online]. Available: https://www.slideshare.net/christian.timmerer/mpegdash-overview-stateoftheart-and-future-roadmap.

[3] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, March 2017.

[4] A. Raake, J. Gustafsson, S. Argyropoulos, M. 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, no. 6, pp. 68–79, November 2011.

[5] Cisco Systems, Inc., "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022," Cisco White Paper, March 2019. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/190320-mobility-ckn.pdf

[6] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, April 2011.

[7] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP - Standards and Design Principles," in *Proceedings of the 2nd ACM Multimedia Systems Conference (MMSys)*, February 2011, pp. 133–144.

[8] P. Juluri and V. Tamarapalli, and D. Medhi, "Measurement of Quality of Experience of Video-on-Demand Services: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, Firstquarter 2016.

[9] *Vocabulary for performance and quality of service. Amd. 2: New definitions for inclusion in Rec. ITU-T P.10/G.100*, ITU-T Recommendation P.10/G.100, July 2008. [Online]. Available: https://www.itu.int/rec/T-REC-P.10/

[10] *Definition of Quality of Experience*, COM 12 - LS 62 - E, TD 109rev2 (PLEN/12), ITU-T Study Group 12, Geneva, Switzerland, January 2007.

[11] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 418–426, June 2005.

[12] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting Multimedia Internet Content for Universal Access," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, March 1999.

[13] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 562–585, Firstquarter 2019.

[14] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?" in *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2012, pp. 9–14.

[15] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *Proceedings of the 2nd ACM Multimedia Systems Conference (MMSys)*, February 2011, pp. 157–168.

[16] M. N. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnström, and A. Raake, "Quality of Experience and HTTP Adaptive Streaming: A Review of Subjective Studies," in *Proceedings of the 6th International Workshop on Quality of Multimedia Experience (QoMEX)*, September 2014, pp. 141–146.

[17] K. Yamagishi and T. Hayashi, "Parametric Quality-Estimation Model for Adaptive-Bitrate-Streaming Services," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1545–1557, July 2017.

[18] D. Zegarra Rodríguez, R. Lopes Rosa, E. Costa Alfaia, J. Issy Abrahão, and G. Bressan, "Video Quality Metric for Streaming Service Using DASH Standard," *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 628–639, September 2016.

[19] 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, December 2015.

[20] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing Effect Sizes of Influence Factors Towards a QoE Model for HTTP Adaptive Streaming," in *Proceedings of the 6th International Workshop on Quality of Multimedia Experience (QoMEX)*, September 2014, pp. 111–116.

[21] 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 *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, August 2011, pp. 362–373.

[22] N. Barman and M. G. Martini, "QoE Modeling for HTTP Adaptive Video Streaming - A Survey and Open Challenges," *IEEE Access*, vol. 7, pp. 30 831–30 859, March 2019.

[23] S. Wei and V. Swaminathan, "Low Latency Live Video Streaming over HTTP 2.0," in *Proceedings of the 24th ACM Workshop on Network and Operating System Support on Digital Audio and Video (NOSSDAV)*, March 2014, pp. 37–42.

[24] Z. Xu, X. Zhang, and Z. Guo, "QoE-Driven Adaptive K-Push for HTTP/2 Live Streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1781–1794, June 2019.

[25] H. T. Le, T. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, "HTTP/2 Push-based Low-delay Live Streaming over Mobile Networks with Stream Termination," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 28, pp. 2423–2427, September 2018.

[26] M. Xiao, V. Swaminathan, S. Wei, and S. Chen, "Evaluating and Improving Push Based Video Streaming with HTTP/2," in *Proceedings of the 26th ACM Workshop on Network and Operating System Support on Digital Audio and Video (NOSSDAV)*, May 2016, pp. 1–6.

[27] M. Xiao, V. Swaminathan, S. Wei, and S. Chen., "DASH2M: Exploring HTTP/2 for Internet Streaming to Mobile Devices," in *Proceedings of the 24th ACM international conference on Multimedia (MM)*, October 2016, pp. 22–31.

[28] H. K. Yarnagula, S. Luhadia, S. Datta, and V. Tamarapalli, "Quality of Experience Assessment of Rate Adaptation Algorithms in DASH: An Experimental Study," in *Proceedings of the 8th International Conference on Communication Systems and Networks (COMSNETS)*, January 2016, pp. 1–8.

[29] P. Casinelli, "The Case for Leveraging the Cloud for Video Service Assurance," IneoQuest Technologies, Inc., IneoQuest White Paper, July 2013. [Online]. Available: https://silo.tips/download/the-case-for-leveraging-the-cloud-for-video-service-assurance

[30] S. D. V. V. Krishna, "Cost Aware Virtual Content Delivery Network for Streaming Multimedia," Master Dissertation, Blekinge Institute of Technology, Sweden, October 2015.

[31] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing: A Key Technology Towards 5G," European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, White Paper 11, September 2015. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[32] K. Dong, J. He, and W. Song, "QoE-Aware Adaptive Bitrate Video Streaming over Mobile Networks with Caching Proxy," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, March 2015, pp. 737–741.

[33] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Helping Hand or Hidden Hurdle: Proxy-Assisted HTTP-Based Adaptive Streaming Performance," in *Proceedings of the 21st IEEE International Symposium on Modelling,*

Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), August 2013, pp. 182–191.

[34] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-Assured 4K Video-on-Demand Delivery Through Mobile Edge Virtualization With Adaptive Prefetching," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, October 2017.

[35] *Dash.js Reference Client Implementation*. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js

[36] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, August 2017, pp. 197–210.

[37] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, April 2016, pp. 1–9.

[38] *Eclipse Jetty*. [Online]. Available: http://www.eclipse.org/jetty/

[39] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment Aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming Over HTTP," in *Proceedings of the IEEE International Conference on Communication Workshop*, June 2015, pp. 1765–1770.

[40] A. Beben, P. Wiśniewski, J. M. Batalla, and P. Krawiec, "ABMA+: Lightweight and Efficient Algorithm for HTTP Adaptive Streaming," in *Proceedings of the 7th International Conference on Multimedia Systems (MMSys)*, May 2016, pp. 1–11.

[41] 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, Thirdquarter 2017.

[42] L. Popa, A. Ghodsi, and I. Stoica, "HTTP As the Narrow Waist of the Future Internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*, October 2010, pp. 6:1–6:6.

[43] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing Cost and Performance for Content Multihoming," in *Proceedings of the ACM SIGCOMM*

Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 2012, pp. 371–382.

[44] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A Case for a Coordinated Internet Video Control Plane," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2012, pp. 359–370.

[45] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, February 2014.

[46] P. L. Callet, S. Möller, and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience," in *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, ser. Version 1.2, Lausanne, Switzerland, March 2013.

[47] *Vocabulary for performance and quality of service. Amendment 5: New definitions for inclusion in Recommendation ITU-T P.10/G.100*, ITU-T Recommendation P.10/G.100, July 2016. [Online]. Available: https://www.itu.int/rec/T-REC-P.10-201607-S!Amd5/en

[48] B. Lewcio, *Management of Speech and Video Telephony Quality in Heterogeneous Wireless Networks*, 1st ed., T-Labs Series in Telecommunication Services, Ed. Springer International Publishing, 2014.

[49] *Subjective Video Quality Assessment Methods for Multimedia Applications*, ITU-T Recommendation P.910, April 2008. [Online]. Available: http://www.itu.int/rec/T-REC-P.910-200804-I

[50] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," IETF RFC 2616, June 1999, [Online] Available: https://tools.ietf.org/html/rfc2616.

[51] M. Belshe, R. Peon, and E. M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," May 2015, [Online]. Available: https://tools.ietf.org/html/rfc7540.

[52] I. Morris, "ETSI Drops "Mobile" From MEC," Light Reading, New York, NY, USA, News Analysis, September 2016.

[53] M. Patel et al., "Mobile-Edge Computing - Introductory Technical White Paper," European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, White Paper, September 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf

[54] R. P. Goldberg, "Survey of Virtual Machine Research," *Computer*, vol. 7, no. 9, pp. 34–45, September 1974.

[55] G. I. Klas, "Fog Computing and Mobile Edge Cloud Gain Momentum," Open Fog Consortium, ETSI MEC and Cloudlets, Technical Report, November 2015. [Online]. Available: https://yucianga.info/?p=938

[56] M. Watson, "HTTP Adaptive Streaming in Practice," in *presented at the ACM Multimedia Systems Conference (MMSys)*, February 2011. [Online]. Available: http://web.cs.wpi.edu/~claypool/mmsys-2011/Keynote02.pdf

[57] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback Control for Adaptive Live Video Streaming," in *Proceedings of the 2nd ACM Multimedia Systems Conference (MMSys)*, February 2011, pp. 145–156.

[58] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, February 2013, pp. 19–24.

[59] J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. M. Batalla, and E. Borcoci, "MS-Stream: A Multiple-Source Adaptive Streaming Solution Enhancing Consumer's Perceived Quality," in *Proceedings of the 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, January 2017, pp. 427–434.

[60] G. Cofano, L. D. Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and Performance Evaluation of Network-assisted Control Strategies for HTTP

Adaptive Streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 3s, pp. 42:1–42:24, June 2017.

[61] E. Ozfatura, O. Ercetin, and H. Inaltekin, "Optimal Network-Assisted Multiuser DASH Video Streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 247–265, June 2018.

[62] N. Bouten, R. D. O. Schmidt, J. Famaey, S. Latré, A. Pras, and F. De Turck, "QoE-driven In-network Optimization for Adaptive Video Streaming Based on Packet Sampling Measurements," *Computer Networks*, vol. 81, pp. 96–115, April 2015.

[63] 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*, vol. 12, no. 2, pp. 28:1–28:24, October 2015.

[64] P. H. Thinh, N. T. Dat, P. N. Nam, N. H. Thanh, H. M. Nguyen, and T. T. Huong, "An Efficient QoE-Aware HTTP Adaptive Streaming over Software Defined Networking," *Mobile Networks and Applications*, vol. 25, pp. 2024–2036, October 2020.

[65] A. Bentaleb, A. C. Begen, and R. Zimmermann, "SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking," in *Proceedings of the 24th ACM International Conference on Multimedia (MM)*, October 2016, pp. 1296–1305.

[66] S. Zhao and D. Medhi, "SDN-Assisted Adaptive Streaming Framework For Tile-based Immersive Content Using MPEG-DASH," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks*, November 2017, pp. 1–6.

[67] C. Zhou, C. W. Lin, and Z. Guo, "mDASH: A Markov Decision-Based Rate Adaptation Approach for Dynamic HTTP Streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, April 2016.

[68] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the ACM Conference*

on Special Interest Group on Data Communication (SIGCOMM), August 2015, pp. 325–338.

[69] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer State is Enough: Simplifying the Design of QoE-Aware HTTP Adaptive Video Streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 590–601, June 2018.

[70] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A Deep Q-Learning Framework for DASH Video Streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, December 2017.

[71] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH System," in *Proceedings of the 3rd ACM Multimedia Systems Conference (MMSys)*, February 2012, pp. 11–22.

[72] R. Belda, I. De Fez, P. Arce, and J. C. Guerri, "Look Ahead: A DASH Adaptation Algorithm," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2018, pp. 1–5.

[73] 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, April 2014.

[74] L. Liu, C. Zhou, X. Zhang, and Z. Guo, "A Fairness-aware Smooth Rate Adaptation Approach for Dynamic HTTP Streaming," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2015, pp. 4501–4505.

[75] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in *Proceedings of the Internet Measurement Conference (IMC)*, November 2012, pp. 225–238.

[76] 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 ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, August 2014, pp. 187–198.

[77] T.-Y. Huang, R. Johari, and N. McKeown, "Downton Abbey Without the Hiccups: Buffer-based Rate Adaptation for HTTP Video Streaming," in *Proceedings of the ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking (FhMN)*, August 2013, pp. 9–14.

[78] L. Xie, C. Zhou, X. Zhang, and Z. Guo, "Dynamic Threshold Based Rate Adaptation for HTTP Live Streaming," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[79] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation Algorithm for Adaptive Streaming over HTTP," in *Proceedings of the 19th International Packet Video Workshop (PV)*, May 2012, pp. 173–178.

[80] J. Kovacevic, G. Miljkovic, V. Mihic, and K. Lazic, "Advanced Rate Adaptation Algorithm in Video Streaming over HTTP," in *Proceedings of the 4th IEEE International Conference on Consumer Electronics (ICCE)*, September 2014, pp. 78–82.

[81] G. Tian and Y. Liu, "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, December 2012, pp. 109–120.

[82] J. He, Z. Xue, D. Wu, D. O. Wu, and Y. Wen, "CBM: Online Strategies on Cost-Aware Buffer Management for Mobile Video Streaming," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 242–252, January 2014.

[83] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, "TFDASH: A Fairness, Stability, and Efficiency Aware Rate Control Approach for Multiple Clients Over DASH," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 198–211, January 2019.

[84] Z. Wang and X. Jiang, "A QoE-Driven Rate Adaptation Approach for Dynamic Adaptive Streaming Over HTTP," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, July 2019, pp. 224–229.

[85] Z. Wang and X. Jiang, "A QoE-Driven Optimization Strategy for Dynamic Adaptive Streaming over HTTP," in *Proceedings of the 27th International Telecommunication Networks and Applications Conference (ITNAC)*, November 2017, pp. 1–7.

[86] A. Elgabli and V. Aggarwal, "FastScan: Robust Low-Complexity Rate Adaptation Algorithm for Video Streaming Over HTTP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2240–2249, July 2020.

[87] D. Suh, I. Jang, and S. Pack, "QoE-enhanced Adaptation Algorithm over DASH for Multimedia Streaming," in *Proceedings of the International Conference on Information Networking (ICOIN)*, February 2014, pp. 497–501.

[88] S. Xiang, L. Cai, and J. Pan, "Adaptive Scalable Video Streaming in Wireless Networks," in *Proceedings of the 3rd ACM Multimedia Systems Conference (MMSys)*, February 2012, pp. 167–172.

[89] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, August 2016, pp. 272–285.

[90] P. K. Yadav, A. Shafiei, and W. T. Ooi, "QUETRA: A Queuing Theory Approach to DASH Rate Adaptation," in *Proceedings of the 25th ACM International Conference on Multimedia (MM)*, October 2017, pp. 1130–1138.

[91] H. Yuan, H. Fu, J. Liu, J. Hou, and S. Kwong, "Non-Cooperative Game Theory Based Rate Adaptation for Dynamic Video Streaming over HTTP," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2334–2348, October 2018.

[92] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, December 2016, pp. 129–143.

[93] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon, "Cross-layer Scheduler for Video Streaming over MPTCP," in *Proceedings of the 7th International Conference on Multimedia Systems (MMSys)*, May 2016, pp. 1–12.

[94] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the 2nd ACM Multimedia Systems Conference (MMSys)*, February 2011, pp. 169–174.

[95] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate Adaptation for Dynamic Adaptive Streaming over HTTP in Content Distribution Network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, April 2012.

[96] J. Kim, Y.-C. Chen, R. Khalili, D. Towsley, and A. Feldmann, "Multi-source Multipath HTTP (mHTTP): A Proposal," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2014, pp. 583–584.

[97] A. Bentaleb, P. K. Yadav, W. T. Ooi, and R. Zimmermann, "DQ-DASH: A Queuing Theory Approach to Distributed Adaptive Video Streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 1, pp. 1–24, March 2020.

[98] A. Elgabli, K. Liu, and V. Aggarwal, "Optimized Preference-Aware Multi-Path Video Streaming with Scalable Video Coding," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 159–172, January 2020.

[99] C. Mueller, S. Lederer, C. Timmerer, and H. Hellwagner, "Dynamic Adaptive Streaming over HTTP/2.0," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, July 2013, pp. 1–6.

[100] S. Wei, V. Swaminathan, and M. Xiao, "Power Efficient Mobile Video Streaming using HTTP/2 Server Push," in *Proceedings of the 17th IEEE International Workshop on Multimedia Signal Processing (MMSP)*, October 2015, pp. 1–6.

[101] S. Baraković and L. Skorin-Kapov, "Survey and Challenges of QoE Management Issues in Wireless Networks," *Journal of Computer Networks and Communications*, vol. 2013, no. 165146, pp. 1–28, March 2013.

[102] 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, no. 2s, pp. 29:1–29:29, May 2018.

[103] *Reference algorithm for computing peak signal to noise ratio of a processed video sequence with compensation for constant spatial shifts, constant temporal shift, and*

constant luminance gain and offset, ITU-T Recommendation J.340, June 2010. [Online]. Available: https://www.itu.int/rec/T-REC-J.340-201006-I

[104] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: from Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[105] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, *Toward A Practical Perceptual Video Quality Metric*, Netflix Technology Blog, June 2016, Accessed on Feb. 10, 2022. [Online]. Available: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[106] *Objective Perceptual Video Quality Measurement Techniques for Digital Cable Television in the Presence of a Full Reference*, ITU-T Recommendation J.144, March 2004. [Online]. Available: https://www.itu.int/rec/T-REC-J.144-200403-I/en

[107] *Objective Perceptual Multimedia Video Quality Measurement in the Presence of a Full Reference*, ITU-T Recommendation J.247, August 2008. [Online]. Available: https://www.itu.int/rec/T-REC-J.247/en

[108] I. de Fez, R. Belda, and J. C. Guerri, "New Objective QoE Models for Evaluating ABR Algorithms in DASH," *Computer Communications*, vol. 158, pp. 126–140, May 2020.

[109] F. Wang, Z. Fei, J. Wang, Y. Liu, and Z. Wu, "HAS QoE prediction based on dynamic video features with data mining in LTE network," *Science China Information Sciences*, vol. 60, no. 4, pp. 1–14, November 2016.

[110] *Parametric Bitstream-based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services Over Reliable Transport*, ITU-T Recommendation P.1203, October 2017. [Online]. Available: https://www.itu.int/rec/T-REC-P.1203-201710-I/en

[111] R. R. R. Rao, S. Göring, P. List, W. Robitza, B. Feiten, U. Wüstenhagen, and A. Raake, "Bitstream-Based Model Standard for 4K/UHD: ITU-T P.1204.3 - Model Details, Evaluation, Analysis and Open Source Implementation," in *Proceedings of the*

12th International Conference on Quality of Multimedia Experience (QoMEX), May 2020, pp. 1–6.

[112] Z. Guo, Y. Wang, and X. Zhu, "Assessing the Visual Effect of Non-periodic Temporal Variation of Quantization Stepsize in Compressed Video," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2015, pp. 3121–3125.

[113] Y. Ou, Y. Xue, and Y. Wang, "Q-STAR: A Perceptual Video Quality Model Considering Impact of Spatial, Temporal, and Amplitude Resolutions," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2473–2486, June 2014.

[114] H. T. T. Tran, T. Vu, N. P. Ngoc, and T. C. Thang, "A Novel Quality Model for HTTP Adaptive Streaming," in *Proceedings of the 6th IEEE International Conference on Communications and Electronics (ICCE)*, July 2016, pp. 423–428.

[115] H. T. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A Multi-Factor QoE Model for Adaptive Streaming over Mobile Networks," in *Proceedings of the IEEE Global Communications Conference Workshops*, December 2016, pp. 1–6.

[116] W. Robitza, M. Garcia, and A. Raake, "A Modular HTTP Adaptive Streaming QoE Model - Candidate for ITU-T P.1203 ("P.NATS")," in *Proceedings of the 9th International Conference on Quality of Multimedia Experience (QoMEX)*, May 2017, pp. 1–6.

[117] *Opinion Model for Video-Telephony Applications*, ITU-T Recommendation G.1070, June 2018. [Online]. Available: https://www.itu.int/rec/T-REC-G.1070

[118] *The E-model: a computational model for use in transmission planning*, ITU-T Recommendation G.107, June 2015. [Online]. Available: https://www.itu.int/rec/T-REC-G.107/

[119] *Opinion Model for Network Planning of Video and Audio Streaming Applications*, ITU-T Recommendation G.1071, November 2016. [Online]. Available: https://www.itu.int/rec/T-REC-G.1071-201611-I/en

[120] C. Alberti, D. Renzi, C. Timmerer, C. Mueller, S. Lederer, S. Battista, and M. Mattavelli, "Automated QoE Evaluation of Dynamic Adaptive Streaming over HTTP,"

in *Proceedings of the 5th International Workshop on Quality of Multimedia Experience (QoMEX)*, July 2013, pp. 58–63.

[121] J. Lievens, A. Munteanu, D. De Vleeschauwer, and W. Van Leekwijck, "Perceptual Video Quality Assessment in HTTP Adaptive Streaming," in *Proceedings of the IEEE International Conference on Consumer Electronics*, January 2015, pp. 72–73.

[122] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld, "Modeling the YouTube Stack: From Packets to Quality of Experience," *Computer Networks*, vol. 109, no. 2, pp. 211–224, November 2016.

[123] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management Workshops*, May 2011, pp. 485–492.

[124] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312–322, September 2004.

[125] X. Wang and T. Kwon and Y. Choi and H. Wang and J. Liu, "Cloud-assisted Adaptive Video Streaming and Social-aware Video Prefetching for Mobile Users," *IEEE Wireless Communications*, vol. 30, no. 3, pp. 72–79, July 2013.

[126] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Quality-Adaptive Prefetching for Interactive Branched Video Using HTTP-Based Adaptive Streaming," in *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*, November 2014, pp. 317–326.

[127] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Bandwidth-Aware Prefetching for Proactive Multi-Video Preloading and Improved HAS Performance," in *Proceedings of the 23rd ACM International Conference on Multimedia (MM)*, October 2015, pp. 551–560.

[128] S. K. Mehr, P. Juluri, M. Maddumala, and D. Medhi, "An Adaptation Aware Hybrid Client-cache Approach for Video Delivery with Dynamic Adaptive Streaming over HTTP," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2018, pp. 1–5.

[129] J. Kim, G. Caire, A. F. Molisch, J. Kim, G. Caire, and A. F. Molisch, "Quality-Aware Streaming and Scheduling for Device-to-Device Video Delivery," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2319–2331, August 2016.

[130] S. Wilk, D. Schreiber, D. Stohr, and W. Effelsberg, "On the Effectiveness of Video Prefetching Relying on Recommender Systems for Mobile Devices," in *Proceedings of the 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, January 2016, pp. 429–434.

[131] N. Master, A. Dua, D. Tsamis, J. P. Singh, and N. Bambos, "Adaptive Prefetching in Wireless Computing," *IEEE Transactions on Wireless Communications*, vol. 15, no. 5, pp. 3296–3310, May 2016.

[132] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot Aware Adaptive Video Streaming Using Deep Reinforcement Learning," in *Proceedings of the 26th IEEE International Conference on Network Protocols (ICNP)*, September 2018, pp. 165–175.

[133] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, February 2014.

[134] J. O. Fajardo, I. Taboada, and F. Liberal, "Improving Content Delivery Efficiency Through Multi-Layer Mobile Edge Adaptation," *IEEE Network*, vol. 29, no. 6, pp. 40–46, November 2015.

[135] M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré, and F. De Turck, "An Announcement-based Caching Approach for Video-on-Demand Streaming," in *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, November 2015, pp. 310–317.

[136] 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 (ICN)*, September 2016, pp. 237–242.

[137] 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, April 2018.

[138] C. Li, P. Frossard, H. Xiong, and J. Zou, "Distributed Wireless Video Caching Placement for Dynamic Adaptive Streaming," in *Proceedings of the 26th ACM Workshop on Network and Operating System Support on Digital Audio and Video (NOSSDAV)*, May 2016, pp. 1–6.

[139] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-Many Matching Games for Proactive Social-Caching in Wireless Small Cell Networks," in *Proceedings of the 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 569–574.

[140] H. A. Pedersen and S. Dey, "Enhancing Mobile Video Capacity and Quality Using Rate Adaptation, RAN Caching and Processing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 996–1010, April 2016.

[141] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative Content Caching in 5G Networks with Mobile Edge Computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, June 2018.

[142] T. X. Tran and D. Pompili, "Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, September 2019.

[143] Songqing Chen, Bo Shen, S. Wee, and Xiaodong Zhang, "Segment-based Streaming Media Proxy: Modeling and Optimization," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 243–256, April 2006.

[144] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering Stable High-Quality Video: An SDN Architecture with DASH Assisting Network Elements," in *Proceedings of the 7th International Conference on Multimedia Systems (MMSys)*, May 2016, pp. 1–10.

[145] J. W. Kleinrouweler and S. Cabrero, and P. Cesar, "An SDN Architecture for Privacy-Friendly Network-Assisted DASH," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 3, pp. 44:1–44:22, June 2017.

[146] M. F. Tuysuz and M. E. Aydin, "QoE-Based Mobility-Aware Collaborative Video Streaming on the Edge of 5G," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7115–7125, November 2020.

[147] X. Yin, V. Sekar, and B. Sinopoli, "Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets)*, October 2014, pp. 1–7.

[148] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, Firstquarter 2015.

[149] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a Predictive Model of Quality of Experience for Internet Video," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, August 2013, pp. 339–350.

[150] *Wondershaper - Traffic Shaping Script*. [Online]. Available: https://packages.debian.org/unstable/net/wondershaper

[151] 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 (MMSys)*, February 2013, pp. 114–118.

[152] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video Streaming Using a Location-based Bandwidth-lookup Service for Bitrate Planning," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 8, no. 3, pp. 24:1–24:19, August 2012.

[153] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd ACM Multimedia Systems Conference (MMSys)*, February 2012, pp. 89–94.

[154] DASH Industry Forum, *Guidelines for Implementation: DASH-AVC/264 Test cases and Vectors*, Technical Report, March 2014. [Online]. Available: https://dashif.org/docs/DASH-AVC-264-Test-Vectors-v1.0.pdf

[155] *YouTube recommended resolution & aspect ratios.* [Online]. Available: https://support.google.com/youtube/answer/6375112?hl=en

[156] S. Xu, S. Sen, Z. M. Mao, and Y. Jia, "Dissecting VOD Services for Cellular: Performance, Root Causes and Best Practices," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, November 2017, pp. 220–234.

[157] D. Juszka and Z. Papir, "A Study on Order Effect in a Subjective Experiment on Stereoscopic Video Quality," in *Proceedings of the 7th International Workshop on Quality of Multimedia Experience (QoMEX)*, May 2015, pp. 1–6.

[158] *Methodology for the Subjective Assessment of the Quality for Television Pictures*, ITU-T Recommendation BT.500-13, January 2012. [Online]. Available: https://www.itu.int/rec/R-REC-BT.500-13-201201-I

[159] R. B. Dean and W. J. Dixon, "Simplified Statistics for Small Numbers of Observations," *Analytical Chemistry*, vol. 23, no. 4, pp. 636–638, April 1951.

[160] D. B. Rorabacher, "Statistical Treatment for Rejection of Deviant Values: Critical Values of Dixon's 'Q' Parameter and Related Subrange Ratios at the 95% Confidence Level," *Analytical Chemistry*, vol. 63, no. 2, pp. 139–146, January 1991.

[161] A. Böhrer, "One-sided and Two-sided Critical Values for Dixon's Outlier Test for Sample Sizes up to n = 30," *Economic Quality Control*, vol. 23, pp. 5–13, April 2008.

[162] Nokia Corporation, "Quality of Experience (QoE) of Mobile Services: Can It be Measured and Improved?" White Paper, 2016. [Online]. Available: https://docplayer.net/25986899-White-paper-quality-of-experience-qoe-of-mobile-services-can-it-be-measured-and-improved.html

[163] J. H. Aldrich and F. D. Nelson, *Linear Probability, Logit, and Probit Models: Quantitative Applications in the Social Sciences*, 1st ed., Thousand Oaks, Ed. SAGE Publications, November 1984.

[164] Video Quality Experts Group, "Video Quality Experts Group (VQEG) Full Reference Television (FRTV) Phase I - Final Report," March 2000. [Online]. Available: https://www.its.bldrdoc.gov/vqeg/projects/frtv-phase-i/frtv-phase-i.aspx

[165] *Parametric Non-intrusive Assessment of Audiovisual Media Streaming Quality*, ITU-T Recommendation P.1201, December 2013. [Online]. Available: https://www.itu.int/rec/T-REC-P.1201/en

[166] Comscore, Inc., "U.S. Online Video Rankings by ComScore," April 2012. [Online]. Available: https://www.comscore.com/Insights/Press-Releases/2012/4/comScore-Releases-March-2012-US-Online-Video-Rankings

[167] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori, "DASH Fast Start Using HTTP/2," in *Proceedings of the 25th ACM Workshop on Network and Operating System Support on Digital Audio and Video (NOSSDAV)*, March 2015, pp. 25–30.

[168] J. Heidemann, K. Obraczka, and J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 616–630, October 1997.

[169] 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, December 2013.

[170] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: Power Optimization for Mobile Videostreaming via Dynamic Cache Management," in *Proceedings of the 20th ACM International Conference on Multimedia (MM)*, October 2012, pp. 279–288.

[171] W. Law, A. C. Begen, and J. Evans, *Common Media Server Data (CMSD): Working Draft*, 2022, Accessed on Feb. 10, 2022. [Online]. Available: https://github.com/cta-wave/common-media-server-data

[172] Q. He, C. Dovrolis, and M. Ammar, "On the Predictability of Large Transfer TCP Throughput," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 145–156, August 2005.

[173] S. Tao and R. Guérin, "Application-Specific Path Switching: A Case Study for Streaming Video," in *Proceedings of the 12th Annual ACM International Conference on Multimedia (MM)*, October 2004, pp. 136–143.

[174] W. Wei, B. Wang, and D. Towsley, "Continuous-time Hidden Markov Models for Network Performance Evaluation," *Performance Evaluation*, vol. 49, no. 1, pp. 129–146, September 2002.

[175] K. Salamatian and S. Vaton, "Hidden Markov Modeling for Network Communication Channels," in *Proceedings of the ACM international conference on Measurement and modeling of computer systems (SIGMETRICS)*, June 2001, pp. 92–101.

[176] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "CFA: A Practical Prediction System for Video QoE Optimization," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI)*, March 2016, pp. 137–150.

[177] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics, M. Jordan, J. Nowak, and B. Schoelköpf, Ed. Springer, February 2006.

[178] ISO/IEC 13818-1:2007, *Generic coding of moving pictures and associated audio information: Systems*, 2007. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:13818:-1:ed-3:v1:en

[179] Consumer Technology Association, *CTA-5004: Web Application Video Ecosystem-Common Media Client Data*, September 2020, Accessed on Feb. 10, 2022. [Online]. Available: https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5004-final.pdf

[180] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, "Common Media Client Data (CMCD): Initial Findings," in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, July 2021, pp. 25–33.

[181] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, November 2016.

[182] *Sprint Network SLA Performance, 2020*, (Accessed on: February 2020). [Online]. Available: https://www.sprint.net/tools/sla-performance/sl

[183] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 Video Delivery over Cellular Networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, October 2016, pp. 1–6.

[184] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim, "MPEG DASH SRD: Spatial Relationship Description," in *Proceedings of the 7th ACM International Conference on Multimedia Systems (MMSys)*, May 2016, pp. 1–8.

[185] N. A. Chinchor, J. J. Thomas, P. C. Wong, M. G. Christel, and W. Ribarsky, "Multimedia Analysis + Visual Analytics = Multimedia Analytics," *IEEE Computer Graphics and Applications*, vol. 30, no. 5, pp. 52–60, August 2010.

[186] S. Pouyanfar, Y. Yang, S.-C. Chen, M.-L. Shyu, and S. S. Iyengar, "Multimedia Big Data Analytics: A Survey," *ACM Computing Surveys*, vol. 51, no. 1, pp. 10:1–10:34, January 2018.

[187] D. A. Adjeroh and K. C. Nwosu, "Multimedia Database Management - Requirements and Issues," *IEEE MultiMedia*, vol. 4, no. 3, pp. 24–33, July 1997.

[188] R. Kosara and J. Mackinlay, "Storytelling: The Next Step for Visualization," *Computer*, vol. 46, no. 5, pp. 44–50, May 2013.

[189] J. Choo and H. Park, "Customizing Computational Methods for Visual Analytics with Big Data," *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 22–28, April 2013.

[190] Y. Cui, T. Li, C. Liu, X. Wang, and M. Kuhlewind, "Innovating Transport with QUIC: Design Approaches and Research Challenges," *IEEE Internet Computing*, vol. 21, no. 2, pp. 72–76, March 2017.

[191] C. Timmerer and A. Bertoni, "Advanced Transport Options for the Dynamic Adaptive Streaming over HTTP," *CoRR*, vol. abs/1606.00264, pp. 1–6, June 2016. [Online]. Available: https://arxiv.org/abs/1606.00264

List of Publications

Publications from Thesis Work:

Refereed Journals

1. **Hema Kumar Yarnagula**, Parikshit Juluri, Sheyda Kiani Mehr, Venkatesh Tamarapalli, and Deep Medhi, "QoE for Mobile Clients with Segment Aware Rate Adaptation Algorithm (SARA) for DASH Video Streaming", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, Volume 15, Issue 2, pp. 36:1-36:23, June 2019.
   DOI: `https://doi.org/10.1145/3311749`

Refereed Conferences

2. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "Score-based Objective Quality of Experience Assessment of DASH Adaptation Algorithms", *10th IEEE International Conference on Advanced Networks and Telecommunications Systems (IEEE ANTS '16)*, IISc Bangalore, India, November 2016. **[Best Paper Award]**

3. **Hema Kumar Yarnagula**, Shubham Luhadia, Soumak Datta, and Venkatesh Tamarapalli, "Quality of Experience Assessment of Rate Adaptation Algorithms in DASH: An Experimental Study", *The Eigth International Conference on COMmunication Systems and NETworkS (COMSNETS '16)*, Bangalore, India, January 2016.

4. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "Adaptive Prefetching at Network Edge to Improve DASH QoE", *21st International Conference on Distributed Computing and Networking (ICDCN '20)*, Kolkata, India, January 2020.

Journal Submission Under Review

5. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "Towards Improved QoE with Adaptive Segment-aware K-Push for Dynamic Adaptive Streaming over HTTP/2", *Computer Networks*.

6. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "Predicting QoE in HTTP Adaptive Streaming Using Parametric Multinomial Logistic Regression Model", *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS)*.

7. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "Getting Content Even Closer: An Adaptive Segment Prefetching Approach for QoE-Improved DASH in Multi-Access Edge Computing Networks", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*.

Publications other than Thesis Work:

1. **Hema Kumar Yarnagula**, Ram K. Vooda, and Venkatesh Tamarapalli, "A Measurement Study of Energy Consumption and QoE Trade-offs for DASH in Mobile Devices", *10th IEEE International Conference on Advanced Networks and Telecommunications Systems (IEEE ANTS '16)*, IISc Bangalore, India, November 2016.

2. Midhul Varma, **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "WebRTC-based Peer Assisted Framework for HTTP Live Streaming", *The Ninth International Conference on COMmunication Systems and NETworkS (COMSNETS '17)*, Bangalore, India, January 2017. **[Best Poster Award]**

3. **Hema Kumar Yarnagula**, Anandi R., and Venkatesh Tamarapalli, "Objective QoE Assessment of DASH Adaptation Algorithms over Multipath TCP", *The Eleventh 11th International Conference on COMmunication Systems and NETworkS (COMSNETS '19)*, Bangalore, India, January 2019.

4. **Hema Kumar Yarnagula**, and Venkatesh Tamarapalli, "The HTTP/2 Server Push and Its Implications on Mobile Web Quality of Experience", *Twenty Fifth National Conference on Communications (NCC '19)*, IISc Bangalore, India, February 2019.

Doctoral Committee

Chairperson:	Prof. Diganta Goswami Professor Department of Computer Science and Engineering Indian Institute of Technology Guwahati
Research Advisor:	Dr. Venkatesh Tamarapalli Associate Professor Department of Computer Science and Engineering Indian Institute of Technology Guwahati
Indian External Examiner:	Prof. Krishna Moorthy Sivalingam Institute Chair Professor Department of Computer Science and Engineering Indian Institute of Technology Madras (IIT Madras), India. *IEEE Fellow, ACM Distinguished Scientist, INAE Fellow.*
Foreign External Examiner:	Prof. Ali C. Begen Professor, Department of Computer Science, Ozyegin University, Istanbul, Turkey. Co-founder, Networked Media. Technical Consultant, Comcast Corporation, USA.
Members:	Dr. Samit Bhattacharya Associate Professor Department of Computer Science and Engineering Indian Institute of Technology Guwahati
	Prof. Arnab Sarkar Associate Professor Advanced Technology Development Centre (ATDC) Indian Institute of Technology Kharagpur [Previously, associated with the Department of Computer Science and Engineering, IIT Guwahati.]
	Dr. John Jose Associate Professor Department of Computer Science and Engineering Indian Institute of Technology Guwahati

Vitae

Hema Kumar Yarnagula joined the Ph.D. programme in the Department of Computer Science and Engineering at Indian Institute of Technology (IIT) Guwahati, India in July 2014. In IIT Guwahati, he was affiliated with the User-Centric Computing and Networking (UCNET) Lab of the Department of Computer Science and Engineering. Prior to joining his Ph.D., he worked as a project associate at the Department of Computer Science and Engineering, Tezpur University, India. He received his Master of Technology (M.Tech) in Information Technology from Tezpur University, Assam, India, in 2013. Prior to that, he received his Master of Computer Applications (MCA) and Bachelor of Computer Application (BCA) degrees from IGNOU, Delhi, India, in 2010 and 2008, respectively. He received the Best Paper Award at the 10th IEEE International Conference on Advanced Networks and Telecommunications Systems (IEEE ANTS 2016) in Bengaluru, India. He also received the Best Poster Award at the 9th International Conference on Communication Systems and Networks (COMSNETS 2017) in Bengaluru, India. He was also awarded the ACM SIGCOMM travel grant and COMSNETS student travel grant to present his papers. His Ph.D. work focuses on Quality of Experience assessment, modeling, and optimization for adaptive multimedia streaming services in mobile environments. His current research interests also include Virtual reality, Virtualized Content Delivery Networks (vCDN), Multi-access Edge Computing, and Multimedia analytics for video surveillance. He enjoys reading books, playing badminton, numismatics, and traveling to new places.

Contact Information

E-mail: h.yarnagula@iitg.ac.in
hemayarnagula@gmail.com

Address: S/o: Kamala Yarnagula, Near Pragati Rice Mill, Yernagula Street,
New Kashinagar, PO/PS: Kashinagar, Gajapati, Odisha - 761206, India