CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Improving video streaming experience through network measurements and analysis
Authors	Raca, Darijo
Publication date	2019-09-10
Original Citation	Raca, D. 2019. Improving video streaming experience through network measurements and analysis. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	2019, Darijo Raca. - https://creativecommons.org/licenses/by-nc-nd/4.0/
Download date	2026-11-09 01:29:25
Item downloaded from	https://hdl.handle.net/10468/9667



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Improving Video Streaming Experience through Network Measurements and Analysis

Darijo Raca
MSc
114220543

Thesis submitted for the degree of Doctor of Philosophy



NATIONAL UNIVERSITY OF IRELAND, CORK

College of Science, Engineering and Food Science

School of Computer Science & Information Technology

September 2019

Head of School: Prof Cormac J. Sreenan

Supervisors: Prof Cormac J. Sreenan
Dr Ahmed H. Zahran

Contents

List of Figures . . . iv
List of Tables . . . viii
Abstract . . . xii
Acknowledgements . . . xiv

1 Introduction **1**
1.1 Overview of 4G cellular networks . . . 2
1.2 HAS performance in cellular networks . . . 3
1.3 Throughput prediction in cellular networks . . . 4
1.4 Interaction between HAS and non-HAS traffic . . . 5
1.5 Summary of thesis contributions . . . 6
1.6 Thesis statement . . . 7
1.7 Thesis structure . . . 7
1.8 List of publications . . . 8

2 Background, Related Work, and Problem Analysis **10**
2.1 Video streaming evolution . . . 11
2.2 Challenges in HTTP adaptive streaming . . . 15
2.2.1 Network resource estimation . . . 15
2.2.1.1 Non-machine learning approaches . . . 17
2.2.1.2 Machine learning approaches . . . 17
2.2.2 Multiple HAS clients coexistence . . . 19
2.2.2.1 Video and other traffic . . . 20
2.2.2.2 Video and network queues . . . 21
2.3 Methods for analysing HAS systems . . . 23
2.3.1 HAS-enabled players . . . 23
2.3.2 HAS encoded video content . . . 24
2.3.3 Bandwidth datasets . . . 25
2.3.4 Quality of experience . . . 27
2.4 Research goals . . . 28
2.4.1 Thesis goals . . . 28
2.5 Conclusion . . . 29

3 Experimental Methodology **30**
3.1 4G dataset . . . 31
3.1.1 Possible use-cases for 4G dataset . . . 32
3.1.2 Dataset collection . . . 33
3.1.3 Dataset overview . . . 37
3.1.3.1 Throughput . . . 38
3.1.3.2 Channel . . . 40
3.1.3.3 Context . . . 41
3.1.3.4 Caveats . . . 43
3.2 Testbed frameworks . . . 44
3.2.1 Testbed for wireless experiments . . . 45
3.2.2 Testbed for wired experiments . . . 46

3.2.3 Key features of frameworks . . . 48
3.2.3.1 Video content . . . 48
3.2.3.2 Implemented HAS algorithms . . . 48
3.2.3.3 Web behavioural traffic models . . . 51
3.2.3.4 Performance metrics . . . 54
3.3 Conclusion . . . 57

4 Design Issues with Throughput Prediction for HAS algorithms 59
4.1 Analysis of traditional bandwidth estimators used in HAS players 60
4.2 Integrating throughput predictions with HAS algorithm . . . 63
4.3 Evaluation . . . 64
4.3.1 Trace classification . . . 65
4.3.2 Accurate predictions . . . 65
4.3.2.1 Traces with high variability . . . 66
4.3.2.2 Traces with low variability . . . 68
4.3.3 Inaccurate predictions . . . 69
4.4 Conclusion . . . 74

5 Empowering Video Players with Machine Learning based Throughput Prediction in Cellular Networks 75
5.1 Analysis of HAS player architecture design . . . 76
5.1.1 Revisiting bandwidth estimation techniques . . . 78
5.2 Throughput prediction via ML . . . 80
5.2.1 Throughput Prediction System Overview . . . 80
5.2.1.1 Collector . . . 81
5.2.1.2 Predictor & Trainer . . . 82
5.2.2 Proposed prediction technique . . . 82
5.2.2.1 Quantile summarization techniques . . . 83
5.2.3 Evaluation of the prediction techniques . . . 84
5.2.3.1 4G Dataset . . . 84
5.2.3.2 Metrics . . . 84
5.2.4 Impact of the Quantile summarization technique . . . 86
5.2.5 Impact of prediction horizon and history length . . . 88
5.2.6 Impact of different mobility patterns . . . 94
5.2.7 Metric contribution to prediction accuracy . . . 96
5.2.8 Mobility impact on prediction accuracy . . . 97
5.3 Video experiments in a controlled environment . . . 99
5.3.1 Throughput prediction module . . . 99
5.3.2 Integrating predicted throughput in HAS algorithm . . . 100
5.3.3 Video QoE models . . . 101
5.3.4 Streaming performance results . . . 102
5.4 Real-Time prediction . . . 106
5.5 Discussion . . . 109
5.5.1 Higher data granularity . . . 109
5.5.2 Network data sources . . . 111
5.5.3 Deriving optimal high-level features . . . 113
5.6 Conclusion . . . 114

6 Impact of Network Queues on video performance 116
6.1 Methodology . . . 118
6.1.1 System model . . . 118
6.1.2 Outline of experiments . . . 119
6.1.2.1 HAS video client . . . 120
6.1.2.2 Video traffic . . . 120
6.1.2.3 Web traffic model . . . 120
6.1.2.4 Active queue management . . . 121
6.1.3 Key performance metrics . . . 121
6.2 Experiments with video traffic . . . 122
6.2.1 Looking beyond queue size . . . 125
6.3 Experiments with mixed traffic . . . 129
6.3.1 Impact of web traffic on video performance . . . 129
6.3.2 Impact of video traffic on web performance . . . 134
6.4 Experiments with heterogeneous RTTs . . . 135
6.5 Discussion . . . 141
6.6 Conclusion . . . 143

7 A solution for experience-oriented adaptive queuing 144
7.1 Methodology . . . 145
7.1.1 Experimental testbed . . . 145
7.1.2 HAS video client . . . 146
7.1.3 Web client . . . 146
7.1.4 Key performance metrics . . . 147
7.2 ENDUE's design . . . 147
7.2.1 Selecting the queuing discipline . . . 148
7.2.1.1 Tuning queue parameters . . . 151
7.2.2 Delay-based adaptive scheduling . . . 152
7.3 FIFO vs CoDel vs ENDUE . . . 158
7.4 Discussion . . . 162
7.5 Conclusion . . . 164

8 Conclusion 165
8.1 Contributions . . . 166
8.2 Future work . . . 168
8.2.1 Extending throughput prediction analysis . . . 168
8.2.2 Further analysis of the network queue impact on user experience . . . 170
8.3 Summary . . . 171

Acronyms 192

List of Figures

2.1 Traditional Streaming with RTSP/RTP/RTCP protocols 11
2.2 HAS Streaming Concepts over TCP transport protocol using chunked video content (different colours represent chunk quality, red = low quality, 235Kbps, yellow = medium quality, 1750Kbps, and green = high quality, 4300Kbps) 13
2.3 HAS System Components for content generation and delivery using adaptive streaming technique 14
2.4 Interaction between Application and Transport layer in HAS client 16
2.5 Buffer-filling and steady states in a HAS session 20

3.1 Time-series of application throughput for static mobility pattern and two different mobile operators 38
3.2 Time-series of application throughput for bus mobility pattern and two different mobile operators 39
3.3 Time-series of application throughput for car mobility pattern and two different mobile operators 40
3.4 Boxplot of CQI vs application throughput (car mobility pattern) for network operator A 41
3.5 Boxplot of CQI vs application throughput (car mobility pattern) for network operator B 42
3.6 GPS coordinate for the all measurement points across Ireland . 43
3.7 The autocorrelation coefficient of throughput (car mobility pattern) 44
3.8 Testbed Architecture 45
3.9 Testbed framework architecture for wired experiments 48
3.10 Simplified Web Behavioural Model 53

4.1 Time-series of instantaneous throughput measured every second 61
4.2 Boxplot of residual error for different bandwidth estimators 62
4.3 Two approaches in feeding prediction to HAS algorithm 64
4.4 EXO: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above bars representing the value of each metric for the no-prediction case*) 67
4.5 ELASTIC: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*) 68
4.6 ARBITER+: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*) 69

4.7 EXO: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*) 70
4.8 ELASTIC: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*) 71
4.9 ARBITER+: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*) 72
4.10 Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (EXO) 72
4.11 Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (ELASTIC) 73
4.12 Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (ARBITER+) . . . 73

5.1 Simplified HAS player architecture and addition of the measurement and prediction modules 77
5.2 Accuracy of different throughput estimation techniques for different prediction horizon values 79
5.3 Comparing ExoPlayer sessions with throughput prediction (green, 12 sec. horizon) and without (blue) 79
5.4 Components, implementation, architectural options and examples of AI-driven throughput prediction system 81
5.5 ARE values as a function of history (horizon = 4 seconds) . . . 86
5.6 ARE values as a function of history (horizon = 8 seconds) . . . 87
5.7 ARE values as a function of horizon (history = 8 seconds) . . . 88
5.8 Prediction horizon analysis 89
5.9 Relative average throughput values across different intervals (real trace) 90
5.10 The average ACF of throughput vs. time lag across all traces . . 92
5.11 Relative variance of throughput values across different intervals (real trace) 93
5.12 ARE for 12s horizon and varying history length 94
5.13 Comparison of ARE for various mobility use-cases (PxF12) . . . 95
5.14 ARE vs. RSRP 98
5.15 Empirical Cumulative Distribution of ARE for two RSRP ranges 98
5.16 ARE accuracy for five different prediction horizons (mixed-mobility ML model) 101
5.17 ARBITER+: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 102

5.18 BOLA-E: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 103
5.19 EXO: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 104
5.20 ARBITER+: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 105
5.21 BOLA-E: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 106
5.22 EXO: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*) 107
5.23 Relative improvement of different QoE metrics in real cellular network with respect to the no-prediction case (EXO algorithm, 4s chunks) 109
5.24 Comparison of ARE for fine-grained data for 250ms and 1s granularity (quantile summarization technique) 110
5.25 Comparison of ARE for device and device+network approach (aperiodic sampling interval) 112
5.26 Comparison between different data representation approaches and ML algorithms for 1-second data granularity 114

6.1 Summary of conducted experiments 120
6.2 Quality of Experience (QoE) metrics across different queue lengths 122
6.3 F-Index across different queue lengths 123
6.4 QoE performance for different queuing disciplines and adaptation algorithms 130
6.5 Video performance in presence of web clients for FIFO queuing discipline (all values are normalised to M_1 scenario values for different QoE metrics) 132
6.6 Video QoE and fairness in mixed traffic scenarios for different queuing disciplines and adaptation algorithms 133
6.7 Web QoE performance in multi-traffic scenario for FIFO queuing discipline 135
6.8 Web QoE performance in multi-traffic scenario for AQM queuing discipline 136
6.9 Queue Length impact on QoE in heterogeneous RTT environment 137

6.10 Queue Length impact on Fairness in heterogeneous RTT environment . . . 138
6.11 QoE performance in heterogeneous RTT environment with AQM mechanisms . . . 140

7.1 ENDUE's simplified architecture . . . 148
7.2 Comparison of PLT between CoDel and FIFO Queuing Discipline in two-queue discipline (equal bandwidth share) . . . 150
7.3 Comparison of PLT for different CoDel's target values in two-queue discipline (equal bandwidth share) . . . 153
7.4 ECDF of average queuing delay for different update intervals (M_1) 155
7.5 ECDF of average queuing delay for different update intervals (M_3) 156
7.6 ECDF of average queuing delay for different update intervals (M_5) 157
7.7 Video performance in presence of web clients for CoDel and ENDUE queuing discipline (*The metrics are normalised to the FIFO case with numbers in white boxes representing the value of each metric for the FIFO case*)) . . . 159
7.8 Comparison of PLT between CoDel and FIFO Queuing Discipline in two-queue discipline . . . 161
7.9 Scatter plot of video and web QoE across multiple scenarios, queuing disciplines and adaptation algorithms . . . 163

List of Tables

2.1 Comparison between HAS- and RTP-based streaming systems . 13

3.1 Collected Metrics 36
3.2 Mobility Patterns 37
3.3 Average and Variation Range of Application Throughput (Mbps) across different mobility patterns and mobile operators 38
3.4 Average and Variation Range of device velocity (kph) across different mobility patterns and mobile operators 42
3.5 Ladder for the average HD encoding rate, resolution and frame rate for the used dataset [QZS16] 49
3.6 Sample trace output from modified dashc 49
3.7 Webpage content parameters [PMT12] 54

4.1 QoE Metrics Notation 65

5.1 Learning Curves for RF algorithm as a function of history interval and 12-second horizon 88
5.2 Throughput stats for various mobility patterns 94
5.3 Feature Importance for PxF4 and P4Fx cases 96
5.4 Throughput for selected traces 100
5.5 QoE Metrics Notation 102
5.6 Android API classes used for collecting radio and throughput metrics 108

6.1 Different Parameter values for experiments 119
6.2 Stall Performance across different queue lengths 122
6.3 Summary results for median representation rate (Kbps) across different bottleneck link capacity values 126
6.4 QoE values across different number of clients (V_i **-** i **HAS clients sharing bottleneck link**) 128
6.5 F-Index across different queuing disciplines and adaptation algorithms 129
6.6 Average instability across different network queue lengths (products of BDP) for heterogeneous RTTs 138
6.7 F-Index across different AQMs for heterogeneous RTTs 139
6.8 Average instability across different AQM mechanisms for heterogeneous RTTs 141

7.1 Different Parameter values for experiments 145
7.2 QoE Metrics Notation for Video and Web clients 147
7.3 Average video QoE score across various M_i scenarios with LOGISTIC and ELASTIC adaptation algorithms 149
7.4 Average video QoE score across various *target* values and M_i scenarios with LOGISTIC and ELASTIC adaptation algorithms . . 152

7.5 Average bandwidth allocation for web and video flows across different update intervals and M_i scenarios 157
7.6 F-Index across various M_i scenarios with LOGISTIC adaptation algorithm (M_5 is omitted as it only has one video client competing for resources) 160
7.7 F-Index across various M_i scenarios with ELASTIC adaptation algorithm (M_5 is omitted as it only has one video client competing for resources) 160

I, Darijo Raca, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Darijo Raca

To my loved ones

Abstract

Multimedia traffic dominates today's Internet. In particular, the most prevalent traffic carried over wired and wireless networks is video. Most popular streaming providers (e.g. Netflix, Youtube) utilise HTTP adaptive streaming (HAS) for video content delivery to end-users. The power of HAS lies in the ability to change video quality in real time depending on the current state of the network (i.e. available network resources). The main goal of HAS algorithms is to maximise video quality while minimising re-buffering events and switching between different qualities. However, these requirements are opposite in nature, so striking a perfect blend is challenging, as there is no single widely accepted metric that captures user experience based on the aforementioned requirements. In recent years, researchers have put a lot of effort into designing subjectively validated metrics that can be used to map quality, re-buffering and switching behaviour of HAS players to the overall user experience (i.e. video QoE). This thesis demonstrates how data analysis can contribute in improving video QoE.

One of the main characteristics of mobile networks is frequent throughput fluctuations. There are various underlying factors that contribute to this behaviour, including rapid changes in the radio channel conditions, system load and interaction between feedback loops at the different time scales. These fluctuations highlight the challenge to achieve a high video user experience. In this thesis, we tackle this issue by exploring the possibility of throughput prediction in cellular networks. The need for better throughput prediction comes from data-based evidence that standard throughput estimation techniques (e.g. exponential moving average) exhibit low prediction accuracy. Cellular networks deploy opportunistic exponential scheduling algorithms (i.e. proportional-fair) for resource allocation among mobile users/devices. These algorithms take into account a user's physical layer information together with throughput demand. While the algorithm itself is proprietary to the manufacturer, physical layer and throughput information are exchanged between devices and base stations. Availability of this information allows for a data-driven approach for throughput prediction. This thesis utilises a machine-learning approach to predict available throughput based on measurements in the near past. As a result, a prediction accuracy with an error less than 15% in 90% of samples is achieved. Adding information from other devices served by the same base station (network-based information) further improves accuracy while lessening the need for a large history (i.e. how far to look into the past). Finally, the throughput prediction technique is incorporated to state-of-the-art

HAS algorithms. The approach is validated in a commercial cellular network and on a stock mobile device. As a result, better throughput prediction helps in improving user experience up to 33%, while minimising re-buffering events by up to 85%.

In contrast to wireless networks, channel characteristics of the wired medium are more stable, resulting in less prominent throughput variations. However, all traffic traverses through network queues (i.e. a router or switch), unlike in cellular networks where each user gets a dedicated queue at the base station. Furthermore, network operators usually deploy a simple first-in-first-out queuing discipline at queues. As a result, traffic can experience excessive delays due to the large queue sizes, usually deployed in order to minimise packet loss and maximise throughput. This effect, also known as bufferbloat, negatively impacts delay-sensitive applications, such as web browsing and voice. While there exist guidelines for modelling queue size, there is no work analysing its impact on video streaming traffic generated by multiple users. To answer this question, the performance of multiple videos clients sharing a bottleneck link is analysed. Moreover, the analysis is extended to a realistic case including heterogeneous round-trip-time (RTT) and traffic (i.e. web browsing). Based on experimental results, a simple two queue discipline is proposed for scheduling heterogeneous traffic by taking into account application characteristics. As a result, compared to the state-of-the-art Active Queue Management (AQM) discipline, Controlled Delay Management (CoDel), the proposed discipline decreases median Page Loading Time (PLT) of web traffic by up to 80% compared to CoDel, with no significant negative impact on video QoE.

Acknowledgements

I would like to express my deep gratitude to Professor Cormac Sreenan and Doctor Ahmed Zahran, my research supervisors. I am grateful for their guidance, their support, and their ongoing belief in my research made it possible for me to achieve all that I could.

I wish to thank my MISL colleagues past and present, Mary, Jason, Brian, Thuy, Yusuf, Ziba, Dapong, Hazzaa, Ilias, Alexander, to name but a few, for their help during my time here.

Special thank you is reserved for my friends Elma and Merim. Elma helped me a lot when I first came to Ireland and she was immense support during these four years. For all difficulties during my journey, Merim was always there, ready to give advice and help. I thank You with all my heart.

I also want to thank my colleague Brian for all the coffee mornings and insightful discussions.

I would also like to thank my roommate. Eoghan, thank you for enduring with me all these years.

Finally, I would like to acknowledge the support provided by the Science Foundation Ireland (SFI).

Chapter 1

Introduction

The number of mobile devices is projected to reach 9.1 billion by 2023, growing from 7.1 billion at present [Eri17]. This follows an 18x growth of cellular data traffic in the last five years, primarily due to video traffic, which is expected to reach 78% of all mobile traffic by 2021 [Cis17]. Furthermore, interactive real-time applications (e.g. VoIP, messaging) account for over 2.6 billion active users per month[1]. While video traffic has high throughput requirements, interactive applications on the other hand require low latency from the network.

Recent years have witnessed a tremendous rise in the use of HAS due to its wide adoption by major video content providers. In HAS video systems, each video file is split into a series of contiguous chunks, each with a duration of several seconds. Each chunk is encoded into a set of different quality representations, having different sizes and corresponding to different average throughput rates. These are stored on a server and fetched in sequence by a client (i.e. video player) using HyperText Transfer Protocol (HTTP), resulting in a flow of video from the server to the client. The HAS client implements an adaptation algorithm that determines the best quality to select for the next segment based on the current operating conditions. The effective design of HAS adaptation algorithms is complex and has been a focus of sustained research [BTB+19].

This thesis focuses on improving HAS user experience by analysing network data and design. HAS performance is analysed in wireless and wired networks, including cases with competing traffic (i.e. other HAS and Web clients). This chapter starts by introducing cellular networks, followed by HAS performance shortcomings in these networks. Motivation for the throughput prediction is presented

[1] https://www.statista.com/topics/1523/mobile-messenger-apps/

next, and benefits of throughput prediction for HAS clients are presented, followed by an introduction of challenges when multiple HAS and non-HAS clients share network resources. Finally thesis contributions are presented followed by an outline of the remaining of thesis.

1.1 Overview of 4G cellular networks

Since the dawn of the first wireless cellular network in the late 70's mobile network evolution has exploded, resulting in capabilities and services beyond the original voice communication design. Forty years later, mobile handsets are part of our everyday routine with a wide variety of use cases, including office related tasks (reading and sending emails, making appointments), text messaging, web browsing, playing games and, consuming multimedia content. Cellular data (4G) accounted for 69% of all mobile traffic in 2016, while 3G accounted for 24%, while cellular speeds grew 3x from an average of 2 Mbps in 2015 to 6.8 Mbps in 2016 [Cis17]. These rates are expected to grow by orders of magnitude when the next iteration of the cellular standard, known as 5G, is deployed in 2020.

Mobile devices (e.g. smartphone) connect to a cellular base station (BS), e.g. evolved Node B (eNodeB) in 4G networks. Each HAS governs mobile devices in two ways. First, it handles signalling information from devices, such as handover commands and channel state information (information about user channel environment). Second, it schedules data transmission by allocating its resources, (e.g. physical resource blocks in 4G), to the connected devices. BSs deploy opportunistic schedulers that compromise the tradeoff between fair and efficient resource allocation [CPG+13]. A commonly used strategy in these schedulers is the proportional-fair algorithm, which utilises a device's physical layer information (measured and sent by the device to eNodeB) together with recent throughput.

However current 4G data throughput rates can fluctuate over a period of a few seconds, due primarily to scheduling decisions at the base station, and sudden changes in the underlying radio channel. These changes are caused by inter-cell interference, congestion due to a number of devices per cell, and location of the device relative to the cell edge. This throughput variation is inherently a part of the underlying communication system since the first wireless networks and will be further exacerbated in 5G due to technical issues such as non-line of sight and a reduction in overall transmission distance. These variations in throughput can limit the user QoE, especially when they cause visible degradation in

viewable quality as can occur while streaming audio or video. Underlying network protocols can mitigate these issues, such as Transmission Control Protocol (TCP) whose design reflects throughput variation by embedding an Exponential Weighted Moving Average (EWMA) statistic to adapt to rate-distortion [For02]. Additionally, adaptation algorithms proposed for HAS [Sto11] can further combat the challenge of consistent quality through buffering and graceful adaptation of video quality. One of the main hurdles for these adaptation algorithms is a lack of a broad cellular dataset that captures these throughput variations, especially when combined with channel and context metrics, on which a solution can be designed and compared with other state-of-art algorithms. Recently, researchers have recognised this problem, which resulted in a number of datasets collected over different wireless technologies and video content datasets [SME17].

Due to the operational aspects of cellular networks outlined, information beyond throughput values, information about channel condition for the client regarding serving eNodeB and neighbouring cells, Global Positioning System (GPS) positions of the client and serving eNodeB, client's speed, and handover events, can aid in decision making for a HAS player and improve overall user experience. Furthermore, all of this information allows a multi-purpose analysis beyond original HAS use cases, such as handover prediction, coverage analysis, mobility prediction. However, such a dataset has been unavailable to the research community.

1.2 HAS performance in cellular networks

Cellular networks are a challenging environment for HAS video streaming due to various reasons. Radio channel conditions and cell load are continuously changing. Data transmission to a mobile device is coordinated by protocols that operate at diverse time scales, e.g. radio channel scheduling at millisecond level vs. congestion control at hundreds of milliseconds to seconds level. Furthermore, the base station scheduler allocates the wireless resources based on the bandwidth demand of each device and their channel conditions; this can cause burstiness in the cellular data traffic. State-of-the-art video clients employ adaptation algorithms that use network and application state to determine the quality of the next video chunk to download. There are several recent algorithms based on approaches such as optimisation [YJSS15a, ZQR+16], control theory [CCPM13], game theory [BBHZ18], machine learning [MNA17], and other heuristics integrating well-known averaging techniques. While these algorithms differ in the

specifics of their decision making, most of them need to know the available network bandwidth to determine the quality of the video to download. Since this information is not readily available to them, clients typically use recent throughput measurements to *estimate* the likely network conditions. This is combined with application state in terms of buffer occupancy and chunk qualities to yield decisions on the quality of future chunks to be selected. The high variability in cellular networks, however, leads to significant throughput estimation errors and in turn results in sub-optimal decisions (see Chapter 4). While there have been proposals that attempt to use cellular-specific information to aid estimation [XZKL16, HZM14], they are in the minority.

1.3 Throughput prediction in cellular networks

Since statistical estimation is affected by the variability in cellular networks, there has been significant interest in exploring the possibility of accurately predicting throughput instead. Indeed, several studies based on trace-driven controlled experiments demonstrate that video streaming quality can be improved in cellular networks if throughput can be predicted accurately [ZEG+15, MTAA+16, RZS+18a]. The main reason is that stalls are avoided because the player would not over-commit to large chunks of high bitrate when throughput drop is predicted [MZA+18]. Additionally, players can avoid streaming the lowest quality after a stall or at startup when high throughput is predicted. Finally, accurate prediction can help reduce the number of quality changes. However, accurate throughput prediction in cellular networks is challenging due to the complex changes and interactions in the network state [ZPC+15]. It is also unclear whether a typical mobile device can extract appropriate network-level information and leverage it to make accurate throughput predictions.

Artificial Intelligence (AI) techniques, especially Machine and Deep Learning (ML/DL), have been successfully applied to several problems in networking, such as cellular traffic forecasting, pattern recognition, and classification [ZPH19]. By examining large amounts of data, these techniques can build sophisticated models, learning patterns that cannot be deduced using traditional statistical data analysis. Machine Learning (ML) and Deep Learning (DL) have been very successful in tackling complex problems that: (1) reflect a pattern, (2) cannot be solved mathematically or described structurally, and (3) have large amounts of example data available. Cellular networks provide data in abundance. Each mo-

bile device measures and collects a wide range of control and data information, some of which it reports to its BS. Each HAS then dynamically schedules its resources based on multitudinous parameters including user environment, traffic demands, device capabilities and system load, leading to noticeable variations in the user throughput. While HAS schedulers are vendor-specific black boxes, the scheduling algorithms represent a collection of predefined steps and actions and thus exhibits consistent behaviour (i.e. for the same inputs, it will produce the same output).

Hence, ML/DL represents an attractive method to extract underlying information and correlations in resource scheduling and make accurate throughput predictions for users based on the abundantly available cellular data. The research challenge is to collect and analyse multi-feature data. Also, how to use data is a challenge that is not trivial to overcome considering that feature engineering is the crucial part that enables ML algorithms to perform with high accuracy illustrating the predictive power of radio metrics. Also, the prediction horizon depends on the HAS application requirements, which is tightly coupled with striking a balance between achieving the highest quality and minimum stalls and switches. Previous studies [SUS16, ZRS18] already show the positive impact of accurate throughput prediction on HAS user experience. Still, implementing a prediction framework on real devices in the real network is missing from the research literature.

1.4 Interaction between HAS and non-HAS traffic

As the volumes of HAS traffic on the Internet increased, specific concerns evolved regarding performance-related interactions between HAS [AABD12] and non-HAS traffic [GN11]. For example, Mansy et al. [MVSA13] show that a HAS client may saturate the network queue at a bottleneck, causing the well-known *bufferbloat* effect for a VoIP client, and explored a client-based approach to alleviate this effect. The focus of related research has been on the design of video adaptation at the client [CCPM13, JSZ14, LZG+14, HJM+14] rather than consideration of the role of network operation. Several recent studies [QZRS15, HG12, ZQRS17] indicated that advanced traffic management techniques, such as rate shaping, could improve streaming performance when multiple video flows share a bottleneck link. However, such techniques require dynamic

changes in routers in response to fine changes in individual HAS flows, raising questions of scalability and efficacy in settings with large numbers of such competing flows.

Furthermore, these techniques rely on information from the network that is hard to obtain and maintain, limiting their widespread deployment in operational networks. AQM techniques take a traffic-agnostic approach in order to provide high utilisation and low delay for competing traffic. These techniques typically manage one queue for all traffic and make the decision for packet marking and dropping based on delay measurements. However, the studies focus on analysing generic TCP traffic flows assuming certain properties (e.g. long- and short-lived TCP flows) measuring application-generic performance metrics, such as utilisation, latency, packet-loss ratio, and fairness. The main limitation of previous studies is in overlooking application-specific characteristics when analysing the performance of cross-traffic sharing a bottleneck link.

The all above-mentioned papers and other studies [HMW15] raise the following research question, *Is it feasible to manage the performance interactions between HAS and non-HAS traffic through changes to the network operation, and specifically the configuration of network queuing, so as to improve the overall quality of experience (QoE)?*. To answer this question, an approach with a focus on application (rather than the network) performance is needed which facilitates a universal solution that does not rely on assumptions that all clients can and would be adapted.

1.5 Summary of thesis contributions

The following is a list of the contributions contained within this thesis:

i. A novel ML-based throughput prediction technique that leverages radio metrics to improve the throughput prediction accuracy in mobile networks significantly. The impact of accurate prediction availability is assessed with three state-of-the-art algorithms showing significant QoE performance metrics improvement when a traditional bandwidth estimators are replaced with a throughput prediction. Finally, a field evaluation is performed in an operational cellular network.

ii. An exhaustive realistic experimental study is performed, yielding several important practical results. 1xBDP rule causes underutilisation in multi-

HAS environment. Also, modern AQM queuing disciplines are ineffective in protecting web clients from competing video traffic.

iii. An effective two-queue discipline is proposed that improves both HAS and web clients' QoE performances when they share network resources. The solution arise from the exhaustive experimental study that analyses the interaction between content (e.g. video rate), link (e.g. capacity, round-trip time) and queue design and its impact on HAS and non-HAS QoE performance.

iv. Tools and techniques that allow gathering and analysing application related data. Also, a 4G dataset is produced with unique information (such as the channel, user and throughput information) and has been released to the research community.

1.6 Thesis statement

This thesis demonstrates that data analysis and machine learning techniques can contribute to improving video experience in cellular networks. In wired networks, queue discipline designed by taking application requirements can improve web user experience in the presence of video traffic.

1.7 Thesis structure

The remainder of the thesis is organised as follows:

- Chapter 2 "Background and Related Work" presents relevant background and related work, as well as an overview of the topics considered and the goals of this work.
- Chapter 3 "Experimental Methodology" presents methodology and testbed used for subsequent chapters.
- Chapter 4 "Design Issues with Throughput Prediction for HAS algorithms" motivates the need for accurate throughput predictions by investigating the impact of ideal and synthetically-induced error predictions on HAS QoE performance in cellular networks.

- Chapter 5 "Empowering Video Players with Machine Learning based Throughput Prediction in Cellular Networks" presents a novel machine learning prediction engine for accurate throughput prediction and quantifies its impact on HAS QoE in a lab-controlled testbed and in a real operational cellular network.
- Chapter 6 "Impact of Network Queues on video performance" presents exhaustive realistic evaluation of multiple HAS and non-HAS clients sharing network resources and quantifies the impact of network queue design on their QoE.
- Chapter 7 "Two-Queue Queuing discipline for Heterogeneous Traffic" presents a simple two-queue scheduling discipline for heterogeneous traffic (HAS and web) that significantly improves the QoE of web traffic while having negligible impact on video QoE compared to state-of-the art AQM disciplines.
- Chapter 8 presents conclusions and future work.

1.8 List of publications

The following list contain publications that emanated from the thesis. The main author was involved in devising the key ideas for the paper, collecting the dataset, and writing the paper. Co-authors contributed in discussion, collection of data and editing the paper.

1. **Darijo Raca**, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello, "Empowering video players in cellular: throughput prediction from radio network measurements." In Proceedings of the 10th ACM Multimedia Systems Conference (MMSys '19).
2. **Darijo Raca**, Yusuf Sani, Cormac J. Sreenan, and Jason J. Quinlan, "DASHbed: a testbed framework for large scale empirical evaluation of real-time DASH in wireless scenarios." In Proceedings of the 10th ACM Multimedia Systems Conference (MMSys '19), Open Dataset & Software Track.
3. **Darijo Raca**, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan, "Beyond throughput: a 4G LTE dataset with channel and context metrics."

In Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18), Open Dataset & Software Track.

4. **Darijo Raca**, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello, "Incorporating Prediction into Adaptive Streaming Algorithms: A QoE Perspective." In Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '18).

5. **Darijo Raca**, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan, "Back to the Future: Throughput Prediction For Cellular Networks using Radio KPIs." In Proceedings of the 4th ACM Workshop on Hot Topics in Wireless (HotWireless '17).

6. **Darijo Raca**, Ahmed H. Zahran and Cormac J. Sreenan, "Sizing Network Buffers: An HTTP Adaptive Streaming Perspective," In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Vienna, 2016, pp. 369-376.

The following list outlines publications emanated from collaboration with fellow researchers:

1. A. H. Zahran, D. Raca and C. J. Sreenan, "ARBITER+: Adaptive Rate-Based InTElligent HTTP StReaming Algorithm for Mobile Networks," in IEEE Transactions on Mobile Computing, vol. 17, no. 12, pp. 2716-2728, 1 Dec. 2018.

2. J. J. Quinlan, D. Raca, A. H. Zahran, A. Khalid, K. K. Ramakrishnan and C. J. Sreenan, "D-LiTE: A platform for evaluating DASH performance over a simulated LTE network," In Proceedings of the 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Rome, 2016, pp. 1-2.

3. A. H. Zahran, J. Quinlan, D. Raca, C. J. Sreenan, E. Halepovic, R. K. Sinha, R. Jana, and V. Gopalakrishnan, "OSCAR: an optimized stall-cautious adaptive bitrate streaming algorithm for mobile networks." In Proceedings of the 8th International Workshop on Mobile Video (MoVid '16).

Chapter 2

Background, Related Work, and Problem Analysis

The past few years has witnessed a tremendous rise in multimedia communication and content sharing over the Internet. In the past, the availability of traditional multimedia was constrained by location and time. The internet together with mobile networks based on Internet Protocol (IP) changed everything by removing the main obstacle for multimedia sharing - availability. However, the Internet is designed for best-effort and non-realtime packet delivery and thus poses a challenge for multimedia distribution/streaming. The best effort packet delivery means that every packet is treated equally, irrespective of context (web, video, voice, etc.). This can deteriorate multimedia application performance, which is susceptible to changes in packet loss, available bandwidth, delay and jitter.

One solution proposed to alleviate this form of multimedia deterioration was introduced in 2005 by Move Networks [BTB+19]. They proposed a mechanism to adapt multimedia content based on the Internet best-effort delivery by introducing adaptive delivery over the HyperText Transfer Protocol (HTTP) protocol.

HTTP adaptive streaming (HAS) has quickly become the dominant technology for delivering video over the Internet due to its adaptation by major content and service providers, like Netflix, YouTube, Hulu, Amazon Prime, HBO GO, and DirecTV.

Section 2.1 outlines a short history of video delivery concepts prior to HAS adoption. The section introduces the main characteristics and components of HAS along with a summary of key differences between HAS and traditional video delivery systems. Section 2.3 provides the *toolchain* for HAS related research. This

includes all necessary tools typically used for supporting testing new HAS ideas and approaches (e.g. new adaptation algorithms). These tools include the HAS-enabled video player, simulation or emulation of the delivery network, traces for representing different bandwidth profiles, HAS-encoded video content and performance metrics for performance assessments. Section 2.2 summarises challenges in HAS-based systems. These challenges can be categorised in two main groups: the challenges arising from the limited power of player visibility of network state and network resources and the interaction of multiple video and non-video clients sharing bandwidth resources. Section 2.4 outlines the research goals of this thesis and Section 2.5 concludes the chapter.

2.1 Video streaming evolution

Before the introduction of the HAS approach, online video content was *pushed* over IP from the media server to the client either by using a connection-oriented protocol, e.g. Real-Time Messaging Protocol (RTMP), or connectionless Real-Time Transport Protocol (RTP).

While the transmission was achieved using the protocols mentioned above, the actual setup of the streaming session is conducted using the Real-Time Streaming Protocol (RTSP). RTSP is a signalling protocol used for establishing and controlling media sessions between client/server. Finally, clients send reports back to the media server using the Real-Time Control Protocol (RTCP). RTCP monitors the Quality of Service (QoS) and provides feedback to the server on the quality of data transmission, allowing the server to perform rate adaptation and data delivery scheduling. Figure 2.1 illustrates the *pushed*-based concept with RTSP/RTP protocols at the heart of it.

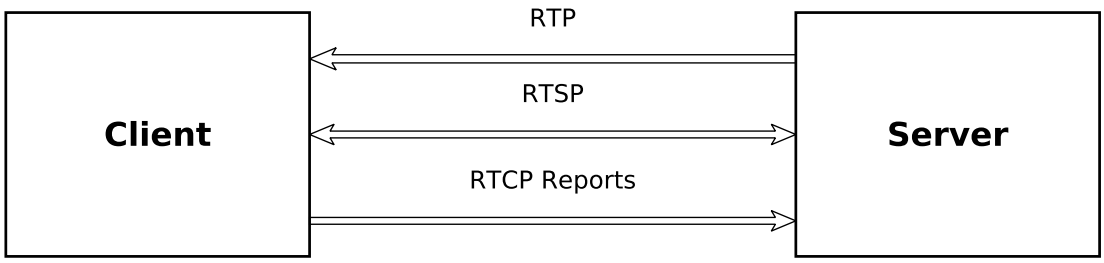


Figure 2.1: Traditional Streaming with RTSP/RTP/RTCP protocols

However, this approach has two main issues. The first drawback of this technology is that media servers are complex and expensive. Also, this approach is not

scalable because all intelligence is at the server (the server has to keep the state for all connections). For a few users and the small quantity of media content, this solution is acceptable. Second, Network Address Translation (NAT) and firewalls are often configured to block signalling RTSP protocol requiring NAT traversal mechanism [GWZ16].

To solve these issues, HAS has taken an inverted approach. Figure 2.2 depicts the *pull*-based concept of the HAS approach. First, intelligence is at the client-side. The client *pulls* media content from the server. Also, clients measure and perform rate adaptation. To adapt to varying network conditions (i.e. change of available bandwidth during playback), HAS partitions media content into multiple smaller files called *chunks* (in literature it is also called *segments*). Each chunk contains video content of some duration. Chunks can be decoded independently of one another. Chunk duration is typically in the range 2 - 10 seconds. Since the clients request chunks independently of one another and the servers do not keep all the state information, this solves the problem with scalability. For the chunk request, the HTTP protocol is used. Transmission Control Protocol (TCP) is the most widely used protocol for transport over HTTP. TCP is a connection-oriented protocol for reliable packet delivery over the Internet. TCP utilises congestion control algorithms. The congestion control algorithm constantly probes the available bandwidth with the goal of the fair share of network resources among competing applications [WDM01]. While TCP is the standard transportation protocol for video delivery, its distinguishing features, reliability (through packet retransmission) and the congestion control (constant rate variation, i.e. well-known *sawtooth* shape), were recognised as the major push for the "anti-TCP" dogma for video streaming in the past [KLW01]. Briefly, packet retransmission may cause unacceptable end-to-end latency, while continuous rate variation (over the short timescale) may cause frequent video quality changes adding to user annoyance. However, for the Video on Demand (VoD) service, latency requirements are less stringent, while the playback buffer and adaptation algorithms can counter rate variation introduced by TCP dynamics [KLW01]. Recently, a new User Datagram Protocol (UDP)-based secure transportation protocol, Quick UDP Internet Connections (QUIC) [LRW+17], was developed with the aim to improve connection setup speeds, reduce latency and enable multiple data connections over the same connection. A combination of HTTP/TCP(QUIC) solves the problem of traversing through NAT and firewalls. Also, it allows usage of conventional HTTP servers and web caches for storing content closer to the end-users.

A HAS session starts by the client requesting the *manifest* file. This file contains

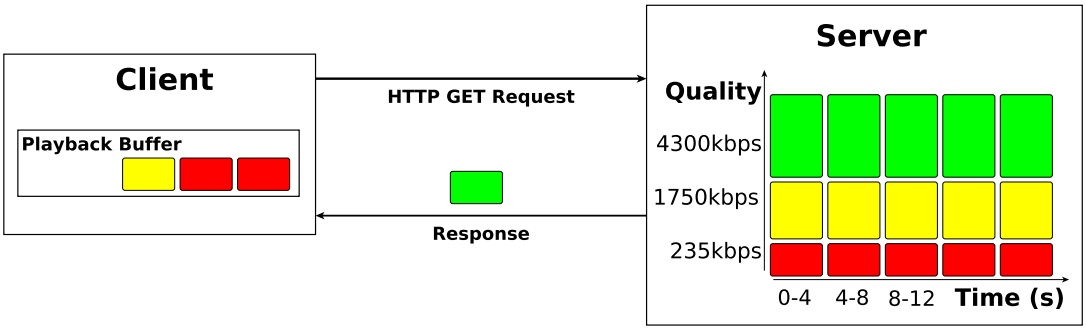


Figure 2.2: HAS Streaming Concepts over TCP transport protocol using chunked video content (different colours represent chunk quality, red = low quality, 235Kbps, yellow = medium quality, 1750Kbps, and green = high quality, 4300Kbps)

all the necessary information about the video content, including the number of chunks, the chunk duration, resolutions and bitrate quality, the addresses of each chunk (URLs), and other features. After parsing the *manifest* file, the client issues HTTP GET request for the chunks. During the download of the chunks, the client measures throughput and monitors the playback buffer levels and other parameters. After downloading a chunk, the client estimates available capacity from the network and/or uses information from the playback buffer to select appropriate bitrate quality for the next chunk.

The client's objective is to stream with the highest bitrate quality possible, while, at the same time, minimising the possibility of stalling (event when the playback buffer is depleted) and switching between different bitrate qualities. However, striking the right balance of these contrasting objectives is not trivial in the process of achieving the highest possible Quality of Experience (QoE) for the user.

Table 2.1 summarises the main aspects of HAS- and RTP-based systems.

Table 2.1: Comparison between HAS- and RTP-based streaming systems

Features	**HAS**	**RTP**
Delivery	Pull-based	Push-based
Protocols	HTTP, TCP, QUIC	RTP, RTSP, RTCP, TCP, UDP
Intelligence at	Client side	Server side
Caching	Yes	Yes, protocol specific

Moving from the HAS server/client request model illustrated so far, we now present an end-to-end content generation and delivery in HAS systems, which typically includes the following components, as depicted in Figure 2.3.

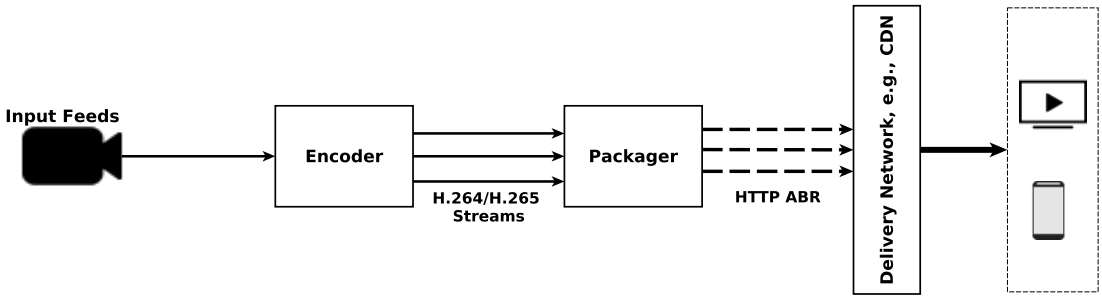


Figure 2.3: HAS System Components for content generation and delivery using adaptive streaming technique

- *Input feeds* represent the source of video content. There are two types of input feeds: pre-recorded for VoD service and live, for live streaming service. For example, streaming services like Netflix and YouTube provide pre-recorded content (i.e. movies, TV shows, podcasts) which are available all the time to the user. Live streaming events (e.g. football matches, live concerts) are recorded in real time during the actual event and sent immediately back to the user.
- *The encoder takes* a raw stream from the input feed and transforms/converts it to a specific digital format. The digital format represents a compressed raw input allowing video content transportation over the Internet. However, compression results in quality degradation, so striking the right balance is crucial. Higher quality will require more storage, driving the need for higher throughput to deliver the content over the network. The format depends on the player's ability to decode encoded content. The specification for compressing and decompressing video (and audio) digital format is called *codec*. The most used video codec for HAS is Advanced Video Coding (AVC)/H.264 [ITU10]. However, newer codecs such as High Efficiency Video Coding (HEVC)/H.265 [SOHW12], AOMedia Video 1 (AV1) [CMH+18], VP9 [FBGP+17] offer up to 70% better compression than H.264. The device support is still limited for these newer codecs slowing down their widespread adoption. Typically, for the HAS system, the encoder generates multiple video/audio encoded streams with different bitrate quality and resolution profiles.
- *Packager* has the role of taking streams from encoder (e.g. H.264 encoded video/audio streams) and packing them for a HAS delivery. This includes fragmenting video content into various bitrate qualities and resolutions. The output of the packager are files ready for delivery over HTTP.

- The role of the *Delivery network* is to deliver the content from the server to the client. The packager provides chunked content to the server. Service providers usually use Content Delivery Networks (CDN) and caching, placing the content closer to the end-users and improving overall performance.
- The role of a client is to send requests for video content, store and decode video chunks. Clients use adaptation algorithms responsible for monitoring network resources and deciding chunk bitrate quality. Adaptation algorithms are the essential component of the client. Over the years, most of the efforts by the research community were centred around improving adaptation algorithms performance.

2.2 Challenges in HTTP adaptive streaming

While the introduction of the HAS systems solves drawbacks of previous server-based streaming approaches (RTP), new problems related to the heterogeneous nature of networks arise, e.g. increasing number of users and demand for high-quality content. [BTB+19].

The HTTP adaptive algorithm is at the heart of the HAS system. Any well designed HAS algorithm should meet the following key objectives:

- High utilisation - efficient use of network resources;
- Fairness - multiple HAS clients should share resources equally;
- Stability - avoid frequent switching which leads to quality fluctuations and stalling, negatively affecting QoE.

2.2.1 Network resource estimation

Adapting to changing conditions in the network depends on the accuracy of the measurement and estimation of available network resources (i.e. throughput). The typical application loop of the HAS client is depicted in Figure 2.4.

The application control loop requests chunk and passes control to TCP. Then the Measurement block monitors chunk download and measures available throughput based on the chunk download time and chunk size. Also, the measurement block monitors buffer level as well, getting an indirect estimate on the throughput.

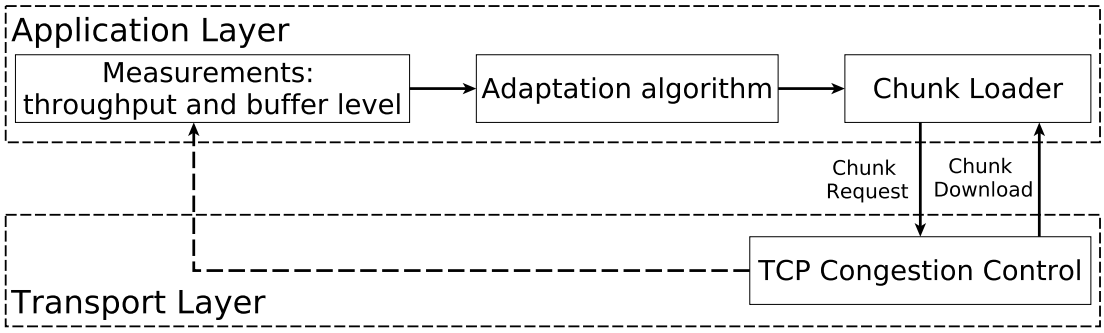


Figure 2.4: Interaction between Application and Transport layer in HAS client

Based on the type of measurement used, adaptation algorithms can accordingly be grouped into three groups: *rate-based* [JSZ14], *buffer-based* [HJM+14, SUS16], and *hybrid* [CCPM13, LZG+14, ZQR+16]. Many adaptation algorithms have been developed over the last decade [KAB17, SME17]. Rate-based approaches use throughput estimates for deciding on the quality for the next chunk. Similarly, buffer-based approaches monitor buffer levels and map it to the chunk quality. However, the third class of algorithms employ a hybrid approach using both throughput estimate and buffer information for a quality decision on the next chunk. There are different approaches and heuristics designed for optimising quality chunk selection. Li et al. [LZG+14] proposed a heuristic inspired by an Adaptive Increase Multiplicative Decrease (AIMD) principle from TCP congestion control for bandwidth estimation. In [CCPM13], the authors apply a proportional-integral-derivative principle with the harmonic mean for driving chunk quality selection. Similar, Yin et al. [YJSS15b] leverage a Model Predictive Control (MPC) optimisation framework and use QoE score as a model objective to optimise chunk rate selection. However, these bandwidth-based adaptation algorithms suffer from low QoE because of frequent stalls and bitrate oscillations, as commonly used bandwidth estimation techniques (e.g. mean, harmonic, median and exponential moving average) exhibit low estimation accuracy.

Out of all environments, cellular networks represent the most challenging environment for throughput estimation and thus HAS clients. Radio channel conditions and cell load are continuously changing. Data transmission to a mobile device is coordinated by protocols that operate at diverse time scales, e.g. radio channel scheduling at millisecond-level vs congestion control at hundreds of milliseconds to seconds level. Furthermore, the base station scheduler allocates the wireless resources based on the bandwidth demand of each device and their channel conditions; this can cause burstiness in the cellular data traffic. For mobile clients that are in motion, the above-mentioned issues are even more exaggerated. As a

result, HAS clients are unable to capture the bandwidth variations in cellular networks. Many studies have shown that existing bandwidth estimation techniques largely over/underestimate the bandwidth share [YXC17, CMK+13, ESS+13].

To solve this issue, researchers propose various throughput prediction techniques. Existing solutions for throughput prediction can be grouped into two general categories: *non-machine learning* and *machine learning* approaches. Furthermore, additional useful categorisation can be made regarding prediction horizon (e.g. short, the order of milliseconds, or medium, the order of seconds) and whether solutions are application-specific (e.g. video or voice).

2.2.1.1 Non-machine learning approaches

Non-machine learning approaches include different techniques for throughput estimation. Lots of work has been done in order to improve TCP estimation performance over cellular networks [WSB13, HRX08a, VKD02, WB13].

Active measurements were also proposed to estimate throughput, round trip time, and packet losses by sending carefully crafted sequences of short data packets [JD03]. Instead of passive measurements, other studies rely on using the device's instantaneous radio Channel Quality Indicator (CQI) and the Discontinuous Transmission Ratio (DTX) for throughput estimation [SYJ+16, LDJ+15].

2.2.1.2 Machine learning approaches

Over the years, applying *machine learning* in throughput forecasting has slowly gained momentum. Most of the work is concerned with improving TCP estimates over shorter horizons [WB13, MSBZ10, XMML13]. Recently, Mei et al. [MHC+19] use Long Short-Term Memory (LSTM) recurrent neural network for a throughput prediction in mobile networks. In their approach, authors use up to 10 seconds of throughput past information to predict up to 5 seconds of future throughput. Authors do not predict average throughput over future horizon; Instead, they predict on a per second basis. For example, for a 5-second future window, model outputs predicted value for each future second. Also, they rely only on information about past throughput and do not leverage any channel information. They compare their solution against Recursive Least Squares (RLS) estimation showing significant improvement in prediction accuracy. Similarly, Yang et al. [YTHL19] applies an RF algorithm for CQI prediction. Authors map predicted CQI value

to achievable throughput. Similar, Samba et al. [SBB+18] test several Machine Learning (ML) algorithms to predict average throughput in the LTE network. Unlike similar approaches, the authors methodology consists of downloading a file of fixed size ($4MB$-$50MB$) and measuring its download rate. Authors analyse different information (channel-related context, such as velocity and distance from the cell, and information from the base station, such as average base station (BS) throughput) and report prediction accuracy depending on which information is used. The introduction of adaptive streaming and the rise of multimedia streaming consumption over cellular networks motivated an investigation of the ability to predict longer horizons. Sayeed et al. use an Autoregressive Integrated Moving Average (ARIMA) based time-series model taking very specific parameters such as Signal to Interference and Noise Ratio (SINR) and Modulation and Coding Scheme (MCS) as inputs to first predict the number of received bits per Physical resource Block (PRB) and then translate that to effective throughput [SGFS15]. Their experiments are evaluated for a stationary device under different channel configurations. Also, some solutions are crafted with a video application in mind. For example, Zou et al. propose an algorithm for HTTP adaptive streaming that relies on an accurate forecast of average throughput [ZEG+15]. Their solution leads to significant improvements in video QoE compared to other state-of-the-art approaches [HJM+14, JSZ14]. In a similar vein, Mangla et al. design an adaptation algorithm that takes prediction errors into account when making a decision for the next chunk [MTAA+16]. Yan et al. [YAZ+19] developed Fugu, a neural network based adaptation algorithm. The algorithm learns continually (by a retraining model based on last week's data) to make more informed predictions for chunk quality selection. Unlike traditional adaptation algorithms that rely on throughput estimation, Fugu uses a probabilistic model and predicts chunk download time instead of throughput. In addition, Fugu outputs a probability distribution for chunk download time and does not rely only on one value. Some solutions look for patterns of similarity between sessions to predict what QoE the new session will have, where similarity is determined through coarse-grained geographic and network features, not precise network performance measurements [SYJ+16]. Xie et al. propose a framework for HTTP adaptive streaming application where authors leverage Long-Term Evolution (LTE) resource structure by monitoring available bandwidth based on PRBs utilisation of the cell, enabling more accurate estimation of available bandwidth. Their approach enables the HAS client to track changes in available bandwidth more accurately resulting in high video quality while minimising stalling rate [XZKL16].

Machine learning has also been used to develop the adaptation logic for video streaming algorithms. In [MNA17], authors propose a reinforcement neural network backed algorithm that learns from real traces the best strategy for adapting to different network conditions.

The literature includes limited studies tackling the throughput prediction problem for video streaming applications using a machine learning approach, considering variable prediction horizons and realistic mobility conditions. Yue et al. [YJS+18] also investigate prediction using only device-level metrics. However, they rely on a UDP based technique for measuring throughput and use the average as their summarization technique. For the horizon, they consider one second. While the framework is based on measurement of radio channel metrics from Android Operating System (OS) they do not quantify its impact on video streaming performance. Furthermore, they compute prediction accuracy using the holdout method (a method where x% of data, e.g. 60%, is used for training, and remaining for testing). However cross-validation is a more reliable method for estimating the model performance. This method is similar to the holdout method but it is repeated multiply times, with different data splits on each iteration. Xie et al. conducted experiments on real mobile devices in a real cellular network [XZKL16]. However, in their approach, they used specialised hardware (Universal Software Radio Peripheral - USRP) for monitoring the wireless channel between device and evolved Node B (eNodeB) and estimating PRB utilisation. Furthermore, all the calculations are done on a laptop which feeds the information back to the device through a USB cable. In real-world experimentation, all the measurements and decisions are done at the mobile device which limits their approach to wide deployment.

2.2.2 Multiple HAS clients coexistence

HAS streaming sessions consist of two phases, buffer filling and steady state [ANBD12] as illustrated in Figure 2.5. In this context, the buffer-filling state (ON) means that the client is downloading a chunk while the steady state (OFF) corresponds to the period when the client must wait to have sufficient space in its buffer to request the next chunk.

Several recent studies, e.g. [AABD12, HG12, HHH+12] pointed out streaming performance issues, such as frequent quality switches and unfairness when multiple video clients share a bottleneck link. The "ON-OFF" behaviour of HAS

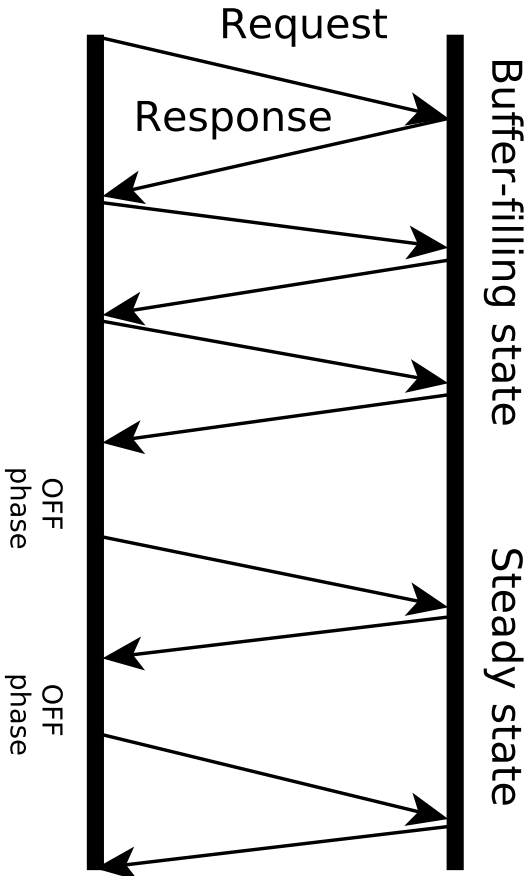


Figure 2.5: Buffer-filling and steady states in a HAS session

clients in the steady state is recognised as the main cause for the aforementioned issues [AABD12]. In [HHH+12], TCP congestion window (*cwnd*) issues are identified as an additional factor, leading to the behaviour recognised as the "downward spiral effect" (a phenomenon where video representation rate drops in the presence of a competing TCP flow), which can be mitigated through the use of larger chunks allowing TCP to have a better estimate of the available bandwidth. However, most state-of-the-art adaptation algorithm's heuristics account for this problem and employ different strategies for mitigating it.

2.2.2.1 Video and other traffic

The impact of video on the performance of other traffic sharing a bottleneck becomes a concern. Video traffic stands accused of triggering bufferbloat phenomena occurring on the Internet [GN11]. This phenomenon occurs when large network buffers get full with video packets causing unnecessarily high latency. This latency negatively affects the TCP congestion avoidance algorithm, creat-

ing a standing queue, when the number of outstanding packets in the network is larger than the Bandwidth Delay Product (BDP) [NJ12]. The number of surplus packets causes the delay, which has a negative impact on interactive applications such as web browsing, VoIP and video conferencing. Mansy et al. [MVSA13] investigated the performance when one video client shares a bottleneck link with one voice client and showed that HTTP adaptive streaming application can cause bufferbloat and harm other delay-sensitive applications sharing the same bottleneck. Similar, Hong et al. [HMW15] analysed the impact on application performance of different traffic types (HAS, VoIP, web, FTP) for different scheduling techniques (including active queue management) in broadband cable networks. The authors performed experiments in ns-2 network simulator.

2.2.2.2 Video and network queues

There are two well known approaches for dimensioning network First In First Out (FIFO) queues, the *rule-of-thumb* [VS94] and the *Stanford rule* [AKM04]. The rule-of-thumb approach specifies that the bottleneck queue size should be equal to the BDP (RTTxC, where RTT is round-time-delay and C is the bottleneck capacity). This rule is often applied at the edge part of the network, where the number of flows and bandwidth capacity is relatively small. The Stanford rule is recommended for a large number of TCP flows (over 250) and very high speed links. Then the recommended router queue size is $BDP/\sqrt{F}$, where F is the number of TCP flows sharing the bottleneck link [AKM04]. Dhamdhere et. al. [DD06] analyse long-lived TCP flows in *ns-2* and its performances. The authors show the benefit of large buffers (2xBDP) compared to tiny buffers (Stanford rule) regarding packet-loss ratio and fairness. However, as pointed out by the authors, implication of buffer sizing on application performance is an open issue. Also, most of the studies argue that full utilisation of long-lived TCP flows can be achieved with a fraction of BDP [AKM04, PDT07, DJD05]. However, prior work has not investigated if these recommended sizes are appropriate when video traffic dominates. HAS exhibits characteristics of both long-lived flows (during ON phase) and short-lived flows (going between ON and OFF phase). Also, compressed video is typically Variable Bit-Rate (VBR) in nature. Hence, HAS operates at a discrete set of rates that vary because of both adaptation decisions and VBR encoding. Adaptive queue management algorithms [Ada13], notably Random Early Detection (RED), are often recognised as a key solution for bounding latency for longer bottleneck queues. However, its complex and tedious

configuration presents a formidable stumbling block, often leading to misconfiguration [NJ12]. Hence, the popular FIFO drop-tail mechanism is still widely used due to its inherent simplicity.

Splitting heterogeneous traffic flows into multiple queues is considered to reduce the impact of traffic interaction. After splitting, various techniques are adopted to handle traffic in these queues, such as Weighted Fair Queuing (WFQ), traffic shaping (excess packets are buffered) or traffic policing (excess packets are dropped) [FPT+16]. Zinner et al [ZJB+14] leverage the Software-Defined Networking (SDN) approach for assessing the impact of dynamic resource allocation for two competing flows (file download and progressive YouTube video stream). Two scheduling techniques were evaluated, namely WFQ and Priority Queuing (PQ). Additionally, different queue lengths were tested and their impact on application throughput, packet drop and duplicate rates. However, in all experiments, competing flows were isolated. Similarly, Hu et al. [HBC+16] propose a Trinity framework for bandwidth shaping in the cloud between competing tenants. They leverage the use of Explicit Congestion Notification (ECN) and priority queuing to achieve bandwidth demands and delay constraints for different users sharing a bottleneck link. Their solution produces minimum bandwidth guarantees for large flows, while at the same time protect short flows (e.g. online web services) from an increased delay. To perform bandwidth shaping at large scale (i.e. scalable) common practice includes moving these operations to end hosts. However, increased CPU and memory usage requires more efficient and scalable solutions [SDV+17].

The majority of these studies focus on analysing generic TCP traffic flows assuming certain properties (e.g. long- and short-lived TCP flows) measuring application-generic performance metrics, such as utilisation, latency, packet-loss ratio, and fairness. The main limitation of previous studies is in overlooking application-specific characteristics when analysing the performance of cross-traffic sharing a bottleneck link. Furthermore, an understanding of the impact of network queuing design (including modern Active Queue Management (AQM) techniques) on application-level metrics is needed, with the focus on using real experiments and traffic. Specifically, the interplay between content, network, and application in various scenarios needs to be analysed.

2.3 Methods for analysing HAS systems

Many challenges and pitfalls arise in HAS-based systems. To analyse and identify many of the problems, researchers typically analyse real systems or build the HAS system in a controlled environment. The most common approach in the community is building a simplified lab testbed. The following components are necessary to support HAS evaluation:

- *HAS video player*: player that is capable of parsing the manifest file, downloading HAS content and optionally decoding and displaying it on the screen.
- *HAS encoded video content*: video content approximates input feed (e.g. source of video content), encoder and packager. This content is encoded and split into chunks with multiple resolutions, and encoders.
- *Bandwidth trace*: summarises network conditions for video content delivery from the CDNs to end-users.
- *QoE model*: used for quantifying HAS performance of different approaches.

2.3.1 HAS-enabled players

Depending on whether the HAS-enabled player decodes and displays video, players can be grouped into three categories. Typical players decode and display video content. There are several players available in the literature. ExoPlayer [1] is a Java based media player for Android OS. GPAC [2] is an open-source multimedia framework written in C. It is one of the oldest players available dating back to 2002. It supports many protocols and standards, including HAS. *dash.js* is a JavaScript multimedia framework for HAS streaming initiated by the Dynamic Adaptive Streaming over HTTP (DASH) Industry Forum. Hybrid players allow for switching between decoding and displaying video content. TAPAS [DCCPM14] is a Python based HAS player with this feature. Emulation players don't decode or display video content. Apart from decoding/displaying, they behave in the same way as regular players. There are several emulation players developed over the past few years. AStream [JTM15] is a Python based emulation HAS player. Similar, dashc [RZS18b] is an Ocaml based emulation HAS player. Due to a lack

[1] `https://github.com/google/ExoPlayer`
[2] `https://gpac.wp.imt.fr/player/`

of decoding and displaying capabilities, these players have lower system requirements allowing researchers to perform experiments with a large number of video clients.

Also, these players vary in their support for experimentation through provided documentation, ease of extending the player with a new adaptation logic, and logging functionalities.

2.3.2 HAS encoded video content

One of the main issues when it comes to testing different adaptation algorithms is the fact that there is a relatively small number of HAS enabled datasets in the video research community. Over time, the content of these datasets has evolved in resolution, encoders and bitrates. Lederer et al. [LMT12], in 2012, released the first publicly available DASH dataset encoded with the H.264 encoder. That dataset consists of 6 Full High Definition (HD) clips encoded in 20 quality bitrates ranging from 50Kbps to 8Mbps. Also the content was encoded in five different chunk duration, 1, 2, 4, 6 10, and 15 seconds. The same authors later released the Distributed DASH dataset [LMT+13], which extended the selection of bitrate to encompass a diverse range of geographical content servers. The datasets of Quinlan et al. [Jas16] provide DASH content with H.265 in addition to H.264 encoded video content. In total 23 clips were encoded with ten quality bitrates, five chunk duration (2, 4, 6, 8, and 10 seconds). Similarly, Zabrovskiy et. al [ZFT18] released content encoded using multiple encoders (H.264, H.265, VP9, and AV1). This dataset consists of 4K content with quality bitrates going up to 20Mbps. 4K datasets with Ultra High Definition resolutions (3840x2160) were generated [LFTP+14, QS18] with 40Mbps maximum quality bitrate.

For video compression, there are two main concepts used in practice: Constant Bit-Rate (CBR) and VBR. CBR encodes video at constant bitrate regardless of the scene complexity. For example, the same bit budget is assigned to a simple scene (low-motion) and a more complex scene (high-motion). As a result, CBR produces video content with varying quality and constant rate. On the other hand, VBR uses the opposite approach, allowing different bit budget for different scene types. For a simple scene, a smaller number of bits is used compared to a complex scene where more bits are used to encode the content. This approach produces varying bitrate with consistent quality. While VBR provides better quality and lowers network requirements (i.e. bandwidth), this is approach has a

more convoluted architecture for encoding, storing and delivering VBR-encoded content [QHP+18].

H.264 is a video compression technology. H.264 is the most widely used codec for HAS. H.264 compresses video content by exploiting redundancy in the signal, i.e. spatial and temporal redundancy. H.264 uses intra-frame prediction (using values of pixels in adjacent parts of the frame to predict values of other pixels) and inter-frame prediction (values of pixels from previous frames in time are a good predictor of current pixels) [WSBL03]. Similarly, H.265 is a successor of H.264 promising 2x better compression than H.264. VP9 is a video codec developed by Google. The main goal of VP9 is to reduce the final video bitrate. VP9 uses progressive encoding and divides the picture in super-blocks of size 64x64 pixels. These blocks can be further divided into 4x4 pixel blocks. This process is called subdivision. Unlike H.265, VP9 supports both horizontal and vertical subdivision [FBGP+17].

2.3.3 Bandwidth datasets

Bandwidth datasets are datasets that contain measurements of available throughput collected over some time period. These datasets can be used to replicate bandwidth characteristics in a controlled environment against different HAS-based solutions, enabling comparison of theirs performance. Existing bandwidth datasets focused primarily on the variance in available bandwidth and typically offered a very limited set of device metrics, such as velocity, Global Positioning System (GPS) and signal strength. Bokani et al. [BHK+16], who offered a dataset, collected from 3G and 4G networks, consists of throughput measurements logged every ten seconds, a timestamp for same and GPS coordinates of the user device itself. The authors utilised a single mobile commute pattern in a metropolitan scenario, and repeated multiple trails within this pattern, warranted by the evidence that network throughput can vary significantly for the same route. They collected a large number of samples across the same path to get statistically significant results on network performance. However, their dataset has a low sampling granularity (ten seconds) and only contains throughput and a very limited set of device values.

Similarly, Xiao et al. and Li et al. collected bandwidth traces over 3G and 4G network respectively [XXW+14, LXW+15]. In both papers, the authors use

MobiNet[3], a custom developed non-rooted android application for downloading content using TCP. The majority of both datasets are collected in high-speed mobility environments (train) where speeds can rise to 310 kph. The content of the datasets consists of information such as application throughput, signal strength, device velocity and eNodeB id. Riiser et al. [RVGH13] obtained bandwidth logs from a 3G network using different mobility patterns; these included tram, train, metro, bus, ferry, and car. The dataset contains a sample granularity in the order of seconds and provides additional information such as timestamp, GPS coordinates of the device, and bandwidth throughput. Also, Hooft et al. [vdHPW+16] used the same approach for collecting 4G network traces for analogous mobility patterns, foot, bicycle, bus, tram, train, and car. However, all these traces focus on acquiring throughput values with high sample granularity. Even though collected in a wireless environment, none of these datasets contain any information about the cellular channel.

The above-mentioned datasets provide sufficient throughput information to evaluate the performance of state-of-the-art HAS algorithms that determine the streamed video quality in response to changes in the operating conditions. However, Xie et al. [XZKL15] use channel information from the wireless channel in addition to the throughput rate to make a more intelligent decision for the next chunk quality. Also, context information such as User Equipment (UE)'s GPS position, velocity, eNodeB GPS position and distance between UE and serving eNodeB can be used for user movement prediction and resource allocation [SBS16]. Wang et al. [WKHC14] utilise these metrics for UE movement and direction prediction to minimise the number of handovers.

To enable the investigation of new approaches that rely on information other than throughput for improving HAS (e.g. combining physical layer information about channel with machine learning algorithms for the throughput prediction) researchers need new datasets with additional information. This opens a space for a collection of new datasets with additional information such as additional channel quality information (e.g. channel quality indicator, reference signal received power etc.), supporting new research ideas for HAS performance improvement.

[3] `http://www.wandoujia.com/apps/thu.kejiafan.mobinet`

2.3.4 Quality of experience

Improving the user perceived QoE is one of the key objective of all solutions. This subjective metric has the highest influence on the design of HAS systems (either at the client side i.e. adaptation logic, or server-side i.e. chunk size, video codec or intermediate point i.e. router configuration). However, deriving QoE is a challenging task as one needs to map objective QoS metrics (average quality, switching frequency, re-buffering events, initial delay) onto subjective QoE scores. Studies show that re-buffering events (stalls) have the highest negative impact on overall user experience. On the other hand, the initial delay up to 16 seconds has a negligible negative effect on QoE [SES+15]. Similarly, switching between different qualities does not have a significant impact on QoE where QoE is mainly driven by quality amplitude itself [HSSZ14]. However, Asan et al. [ARhM+17] show that switching between different resolutions can affect user experience. All these metrics are mutually interdependent, e.g. increasing quality level increases the probability of stall, while low quality leads to low QoE. Striking optimal equity between those metrics represents a formidable task. As a result, a number of QoE models for HAS were developed [DDR13, LDU+15]. These models are subjectively derived and validated either in a lab-controlled environment or with the crowd-sourced approach. Recently, standardised ITU-T Rec. P.1203.1 video QoE model [RGR+17] was published. The P.1203.1 QoE model is derived from subjective studies. It uses MOS (mean opinion score, 5-point scale) to capture user experience of the streaming session, taking into account both video and audio impairments. The main drawback of the standardised model is its limitation to H.264/AVC encoding content to a maximum resolution of Full HD (1920x1080).

Unlike video QoE, for the most part, web QoE is dominated by the delay component. To capture user experience several metrics have been proposed for capturing delay influence on overall QoE. This delay is defined as the User Perceived Page Loading Time (UPPLT). OnLoad is the most widely used metric in literature for measuring UPPLT [BDCR16], and represents an elapsed time between sending a request and loading all objects on the web page. However, perceived Page Loading Time (PLT) might only reflect the loading of the visible area of a web page in which case OnLoad overestimates observed PLT. Similarly, objects can continue loading after onLoad finishes causing underestimation [VBNP16]. Several other metrics were proposed to alleviate these limitations [Inc, BDCR16]. Varvello et al. [VBNP16] developed an EYEORG, a crowdsourcing framework for measuring and capturing UPPLT and web QoE. Surprisingly, onLoad shows

the highest correlation (0.85) with the UPPLT, outperforming more sophisticated PLT metrics.

2.4 Research goals

This thesis explores different approaches and techniques for improving video QoE in cellular and wired networks.

2.4.1 Thesis goals

The thesis goals are as follows:

- Improve video QoE by improving throughput estimation in HAS players;

  Classical throughput estimation techniques produce large prediction errors that can force algorithms to make wrong decisions. This is especially true in highly variable environments, such as cellular networks. These environments exhibit non-linear variations in throughput that linear estimators do not capture.

- Assess the error in throughput prediction values and its impact on HAS QoE-related metrics;

  HAS systems operate on a discrete set of bitrates. This can lessen the need for highly accurate throughput prediction. For example, there is a 30% difference between subsequent rates. This means that predicting any value in that range would lead to the same algorithm decision.

- Implement and evaluate the proposed solution in an operational network;

  Assessing the proposed solution in an operational network enables identifying challenges and limitations of deploying throughput prediction backed HAS player in practice.

- From the network queuing perspective, assess and identify key interactions when multiple HAS and non-HAS players share a bottleneck link;

  Resource sharing between HAS players and delay-sensitive applications (such as web traffic) need exhaustive experimentation to identify all the key interactions from the perspectives of network queues.

- Design a new queuing discipline for improving QoE of both HAS and non-HAS applications.

  Identified interactions between HAS and non-HAS players laydown a foundation for designing a new queuing discipline for improving performance of both HAS and non-HAS applications.

2.5 Conclusion

HTTP adaptive streaming solved many issues plaguing traditional streaming approaches such as scalability, availability, network efficiency, and quality of delivered content. However, HAS performance is still limited, especially in an environment with rapid throughput fluctuations (i.e. wireless) and when multiple HAS, and non-HAS clients share a bottleneck link. The following chapters introduce solutions to these problems by exploring the possibility of improving throughput estimation in cellular networks by applying ML techniques (Chapter 4, and 5). Furthermore, an exhaustive analysis is conducted in the realistic environment with multiple HAS and non-HAS clients, identifying challenges from the network perspective (Chapter 6). Finally, a solution is proposed for improving the quality of experience for both HAS and non-HAS clients in a shared environment (Chapter 7).

Chapter 3

Experimental Methodology

This chapter introduces tools and methodology for analysing performance of HTTP adaptive streaming (HAS)-based system conducted in Chapters 4, 5, 6, and 7. The following chapters use the presented tools for analysing existing and proposing new schemes for the improvement of HAS performance.

There are four crucial components when analysing the performance of HAS-based systems. These are: video content for streaming (sufficiently large dataset of mixed content - resolutions, encoders and genres), bandwidth traces, a range of HAS algorithms to compare against (with different demands and objectives), and objective Quality of Experience (QoE) model (standardised if possible) for performance evaluation.

There exist a few HAS enabled video datasets (Section 2.3.2) in the literature encoded in various quality bitrates, resolutions and codecs. Over the last decade, many HAS algorithms were developed with different approaches in designing their adaptation logic (Section 2.2.1). Similarly, there is a limited set of bandwidth traces collected over a wireless channel (Section 2.3.3). Bandwidth traces collected over the wireless channel are more compelling for the research community. Characteristics of the wireless channel can result in more frequent throughput variations than in wired, where throughput is much more stable. The main aim of all HAS-based systems is to achieve the highest QoE irrespective of whether the adaptation is client-focused, a network focused or distributed between both. Because of its subjective nature, QoE is typically difficult to measure and generalise. To help alleviate this problem, researchers have proposed various QoE models over the years, mapping different streaming video QoE-related metrics (e.g. average quality, switching, stalls) to a QoE score, typically in the range (0-5).

While most of these models only take into consideration a subset of the mentioned QoE-related metrics to find an "optimal" blend between them, some efforts have been made in proposing subjectively-validated QoE models (Section 2.3.4).

This chapter presents all these components in detail. Also, this chapter presents two contributions:

- 4G wireless bandwidth traces including channel metrics
- Two HAS-enabled testbeds for wired and wireless experiments

Previous datasets in this area, focused primarily on the variance in available bandwidth and typically offered a very limited set of device metrics, such as velocity, Global Positioning System (GPS) and signal strength. For comparison of HyperText Transfer Protocol (HTTP) adaptive algorithms, information about bandwidths is sufficient, as most of the algorithms logic rely on estimating throughput values from chunk downloads. However, as explained in detail in Chapters 4, and 5, using additional information about user environment and context can significantly improve user QoE.

Currently, to evaluate their proposed solutions, researchers need to create a framework and numerous state-of-the-art algorithms. Often, these frameworks lack flexibility and scalability, covering only a limited set of scenarios. To fill this gap, this chapter introduces highly customisable real-time frameworks for testing HAS algorithms in both, wired and wireless environment.

3.1 4G dataset

Current 4G data throughput rates can fluctuate over a period of few seconds, due primarily to scheduling decisions at the cell tower, and sudden changes in the underlying radio channel [ZEG+15, MTAA+16, YJS+18]. These changes are caused by inter-cell interference, congestion due to the number of devices per cell, and location of the device relative to the cell edge. This throughput variation is inherently a part of the underlying communication system since the first wireless networks and will be further exacerbated in 5G due to technical issues such as non-line of sight and a reduction in overall transmission distance. These variations in throughput can impact the user QoE, especially when they cause visible degradation in quality, which can occur while streaming video. Underlying network protocols can mitigate these issues, such as Transmission Control Protocol

(TCP) whose design reflects throughput variation by embedding an Exponential Weighted Moving Average (EWMA) statistic to adapt to rate-distortion [For02]. Additionally, adaptation algorithms proposed for HAS [Sto11] can further combat the challenge of consistent quality through buffering and graceful adaptation of video quality. One of the main hurdles for these adaptation algorithms is a lack of a broad cellular dataset that captures these throughput variations, especially when combined with channel and context metrics, on which a solution can be designed and compared with other state-of-the-art algorithms. Recently, researchers have recognised this problem, which resulted in a number of datasets collected over different wireless technologies and video content datasets [SME17].

This chapter presents a dataset collected from real 4G production networks. The production dataset, contains traces from two major Irish mobile operators, with different mobility patterns (static, pedestrian, car, bus and train). While the dataset can be used for comparison of various HAS streaming approaches, information captured in this dataset allows for broader analysis beyond adaptive video streaming research. In addition to throughput values, the dataset includes information about channel condition for the client in respect to serving evolved Node B (eNodeB) and neighbouring cells, GPS positions of the client and serving eNodeB, client's speed, and handover events. All of this information allows a multipurpose analysis beyond our original HAS use cases, such as handover prediction, coverage analysis, mobility prediction etc. This dataset is the first publicly available dataset that contains throughput, channel and context information for 4G networks and is freely available to the research community. The dataset can be found at the following link: `http://www.cs.ucc.ie/~dr11/4G_Dataset/`.

3.1.1 Possible use-cases for 4G dataset

This section outlines some of the possible use-cases for the dataset. Starting with the HAS algorithms, the dataset enables the comparison of different algorithm strategies depending on the information they require for optimisation of chunk selection. Most algorithms calculate on throughput samples only, with some of them requiring finer granularity than chunk duration. However, going beyond the throughput requirement, new strategies mandate channel and context information, allowing them to make more accurate throughput prediction. The proliferation of Commercial Virtual Reality (VR) technology is increasing download demands and is a distinct candidate for evaluation using our dataset. Although VR typically uses progressive download, it is expected that VR will

switch to HAS mechanism in the near future [QJHG16]. This switch will result in the need for designing new adaptation algorithms suitable for VR specific needs (adapting the quality level of tiles).

Another use-case would be handover analysis and prediction. The handover procedure is crucial in cellular networks as it allows continuous connection across different eNodeBs. There are various mechanisms and approaches for handover prediction [GWZ+09, BSJ+11, LFCZ11]. To benefit these approaches, the dataset contains information about handover events and also information about GPS position of the current cell and device, channel metrics for the serving and the neighbouring cell. Finally, generating new bandwidth traces based on the existing traces is an exciting and demanding challenge, as multidimensional statistical analysis is needed over all available metrics. For this task, one approach could be leveraging machine learning techniques. Finally, generating new bandwidth traces based on the existing traces is an exciting and demanding challenge, as multidimensional statistical analysis is needed over all available metrics. For this task, one approach could be leveraging machine learning techniques. For example, Generative Adversarial Networks (GANs) [GPAM+14] can be used to learn high-dimensional probability distributions and provide synthetic bandwidth traces with the same distribution as the original traces. As a result, a large number of realistic traces would be generated and thus relieving researchers of manually collecting vast amounts of network traces, which can be a very tedious task.

3.1.2 Dataset collection

For the production dataset collection, the Android device G-NetTrack Pro mobile network monitoring tool[1] is used. This tool enables the capturing of various channel-related metrics, context-related metrics, downlink and uplink throughput, and also cell-related information. The main advantage of this application is that it does not require a rooted phone. In contrast to G-NetTrack, Li et al. developed an open-source software tool MobileInsight [LPY+16] that can capture radio information directly from the chipsets in real-time. However, the software requires a rooted mobile phone and works with Qualcomm System on Chip (SoC) only. This tool is similar to proprietary Qualcomm's QXDM[2] diagnostic software. While the non-rooted aspect of G-NetTrack is beneficial, there are several limita-

[1] http://www.gyokovsolutions.com/
[2] https://www.qualcomm.com/

tions to the application. First, the minimum granularity of the collected samples is one second. Second, the tool uses the standard Android library (*telephony* class) for reporting channel metrics. Implementation of these callback functions depends on the manufacturer of the mobile SoC chipsets. Also, not all parameters are reported for different cellular technologies (2G/3G/4G). For our dataset, we test mobile devices from three major mobile chipsets manufactures, Qualcomm (*Snapdragon*), Samsung (*Exynos*) and Huawei (*Kirin*). Ultimately, the mobile device chosen is a Samsung J5, which provides a means of capturing all 4G network metrics.

The production dataset has 135 traces capturing various mobility patterns across two major Irish operators, with different data limit caps. The first provider (*operator A*) gives unlimited 4G data, while the second provider (*operator B*) offers only 15GB per month. However, the second operator provides 60GB on social media, including Youtube streaming. For the first mobile operator, a file is continuously downloaded (connection-oriented, TCP) with an average duration of 15 minutes per trace (with a five-second pause after the download completes). A similar approach is used for the second operator, but once the data cap is reached, the procedure is changed by downloading content from Youtube. A URL is generated for the video from Youtube to exploit the higher data cap for social media. For each trial, regardless of the measurement approach, a large file is used ($>$ 50MB) to allow the TCP sending window to ramp up to the maximum size. For the congestion control algorithm, TCP Cubic [HRX08b] is used (a default TCP congestion algorithm in Android OS). As stated, every sample is logged with one-second granularity. As a result, the average trace duration is 15 minutes.

To provide a fair comparison between operators, measurement trials are performed for both operators at the same time (same mobile device model is used to limit the impact of device hardware on throughput rate and channel metrics). This subset of traces permits comparison of mobile operators performances across different parameters (throughput and channel metrics). Competing tests use the same download approach for both cellular operators (file or video download).

The following Table 3.1 outlines the various metrics within production dataset (note that channel metrics are part of 4G standard):

Metric	**Description**
Timestamp	Timestamp of sample

Longitude and Latitude	GPS coordinates of mobile device
Velocity (kph)	Velocity of mobile device
Operatorname	Cellular operator name (anonymised)
CellId	Serving cell for mobile device
NetworkMode	Mobile communication standard (2G/3G/4G)
RSRQ (dB)	Reference Signal Received Quality represents a ratio between Reference Signal Received Power (RSRP) and Received Signal Strength Indicator (RSSI). Signal strength (signal quality) is measured across all Resource Elements (REs), including interference from all sources
RSRP (dBm)	Reference Signal Received Power represents an average power over cell-specific reference symbols carried inside distinct RE. RSRP is used for measuring cell signal strength-/coverage and therefore cell selection
RSSI (dBm)	RSSI represents a received power (wideband) including a serving cell and interference and noise from other sources. Reference Signal Received Quality (RSRQ), RSRP and RSSI are used for measuring cell strength/coverage and therefore cell selection (handover)
Signal to Noise Ratio (SNR) (dB)	Value for Signal-to-Noise Ratio

Channel Quality Indicator (CQI)	Channel Quality Indicator of a mobile device. CQI is feedback provided by User Equipment (UE) to eNodeB. It indicates data rate that could be transmitted over a channel (highest Modulation and Coding Scheme (MCS) with a Block Error Rate (BLER) probability less than 10%), as the function of Signal to Interference and Noise Ratio (SINR) and UE's receiver characteristics. Based on UE's prediction of the channel, eNodeB selects an appropriate modulation scheme and coding rate
DL_bitrate (Kbps)	Download rate measured at the device (application layer)
UL_bitrate (Kbps)	Uplink rate measured at the device (application layer)
State	State of the download process. It has two values, either I (idle, not downloading) or D (downloading)
NRxRSRQ & NRxRSRP	RSRQ and RSRP values for the neighbouring cell
Cell_Longitude & Cell_Latitude	GPS coordinates of serving eNodeB. OpenCellid[3], is the largest community open database that provides GPS coordinates of cell towers
Distance (m)	The distance between the serving cell and mobile device

Table 3.1: Collected Metrics

[3] `https://opencellid.org/`

4G measurement trials are performed (unless otherwise stated) across six different mobility patterns summarised in Table 3.2.

Table 3.2: Mobility Patterns

Type	**Summary**
Static	Static trials (indoor)
Pedestrian	Walking trials around Cork city, Ireland
Bus	Trials include urban and suburban cases
Car	Trials include urban and suburban scenarios
Train	Travelling between Cork - Dublin (240km) and Cork - Farranfore (75km). Combination of 3G and 4G.

3.1.3 Dataset overview

This section gives a short overview of the dataset. Traces are categorised as commute traces as we collected the majority of traces during morning and evening hours while going from home to work and back, and begin with an overview of our trace models:

- *Static:* As the name implies, these traces were collected indoors with mobile devices being stationary. This scenario represents how people typically tend to use their smart devices. Characteristic for the majority of static traces is that throughput is quite stable with relatively low variations compared to mobile traces.
- *Pedestrian:* Outdoor traces while walking around Cork city centre using several different routes. Characteristics of collected traces (average rate and standard deviation) are similar to the static case with slightly more variation due to channel condition and handovers.
- *Bus:* Bus traces using public transport around Cork city. Traces are gathered during weekdays and at the weekends to capture different congestion patterns.
- *Car:* Car traces over the city and suburban routes. This sub-category of a dataset contains the most traces.
- *Train:* Majority of the train traces are a mixture of 3G and 4G for both operators, due to the availability of 4G within major urban areas only.

Following sections provide a more detailed overview of the Throughput, Channel and Context information provided in the dataset: Figure 3.1, 3.2, and 3.3 illustrate a time-series of application throughput for both network operators across static, train and car mobility pattern setups, respectively (randomly selected competing traces are shown).

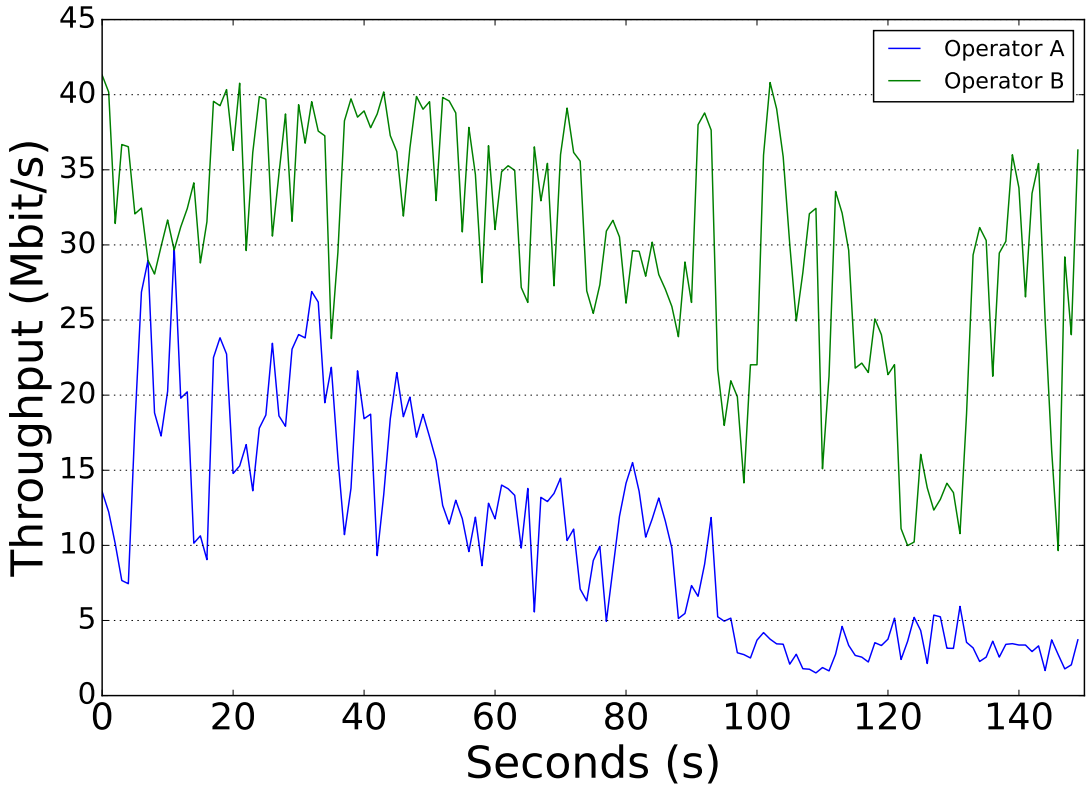


Figure 3.1: Time-series of application throughput for static mobility pattern and two different mobile operators

3.1.3.1 Throughput

Table 3.3: Average and Variation Range of Application Throughput (Mbps) across different mobility patterns and mobile operators

	Mobility Patterns									
Operator	**Static**		**Pedestrian**		**Bus**		**Car**		**Train**	
	Avg.	*Var. Range*	*Avg.*	*Var. Range*	*Avg.*	*Var. Range*	*Avg.*	*Var. Range*	*Avg.*	*Var. Range*
A	5.3	(0.9, 9.3)	9.9	(0.4, 28.0)	8.0	(0.08, 20.3)	11.4	(0.92, 27.9)	4.7	(0, 11.3)
B	42.6	(21.3, 77.2)	18.2	(5.6, 34.2)	13.5	(2.0, 29.1)	22.3	(3.2, 49.1)	6.6	(0.3, 16.5)
Num. Traces	15		31		16		53		20	
Trace Dur. (mins)	254		560		180		1265		650	

Furthermore, Table 3.3 depicts average application throughput and variation, including the number of traces and total trace duration across all traces for

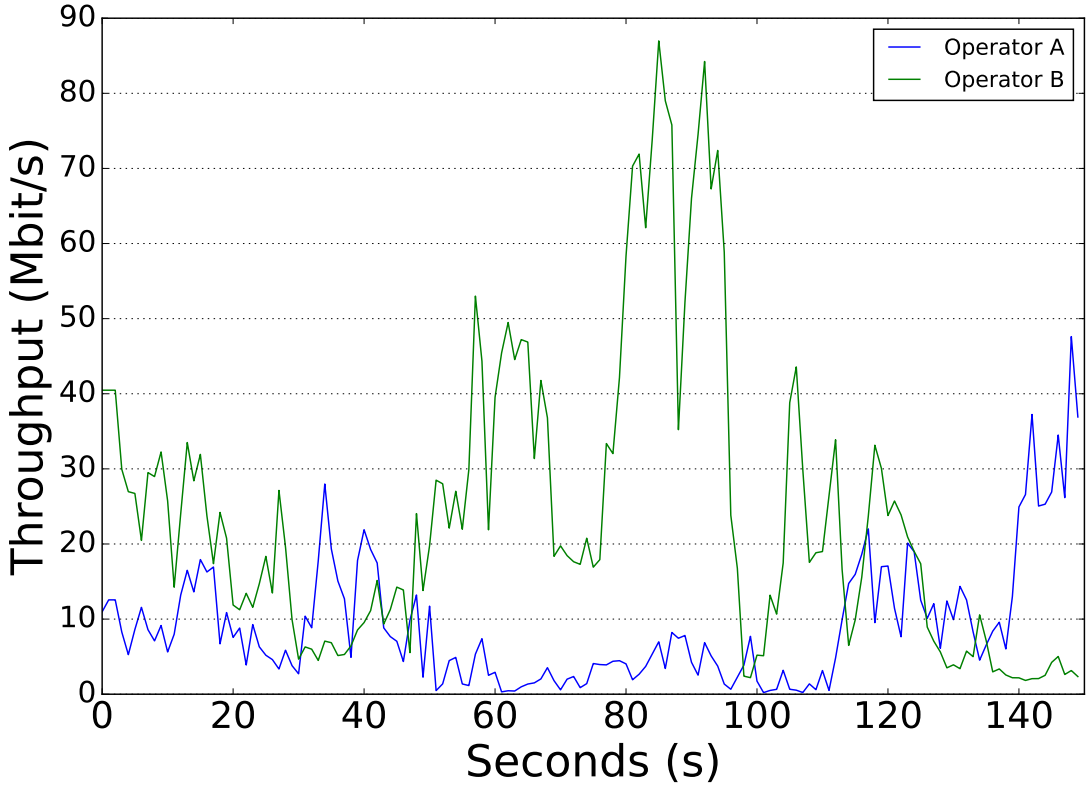


Figure 3.2: Time-series of application throughput for bus mobility pattern and two different mobile operators

different mobility pattern categories and two mobile operators. By definition, variation range is a percentile-wise measure of variation. Let's define R as application throughput during time interval the $(t, t+1)$. Then we can define variation range as the interval $[R^L, R^H]$, where R^L represents a 10^{th} percentile of R, and analogously R^H a 90^{th} percentile of R [JD05]. This range defines boundaries where 80% of measured throughput lies. From the values shown in Table 3.3, operator B has a significantly higher average than operator A for all mobility pattern cases. There could be different reasons for this observation, including better coverage, and the operator's internal traffic policy (e.g. traffic limitation and shaping). Looking at each case individually, there are different changes in average value and variation range depending on the operator itself, e.g. for A, a static case has a significantly lower average than the pedestrian case. A rationale for this result could be in coverage discrepancy for indoor and outdoor scenarios. We note that experiments run indoor have a weaker signal in 90% of cases.

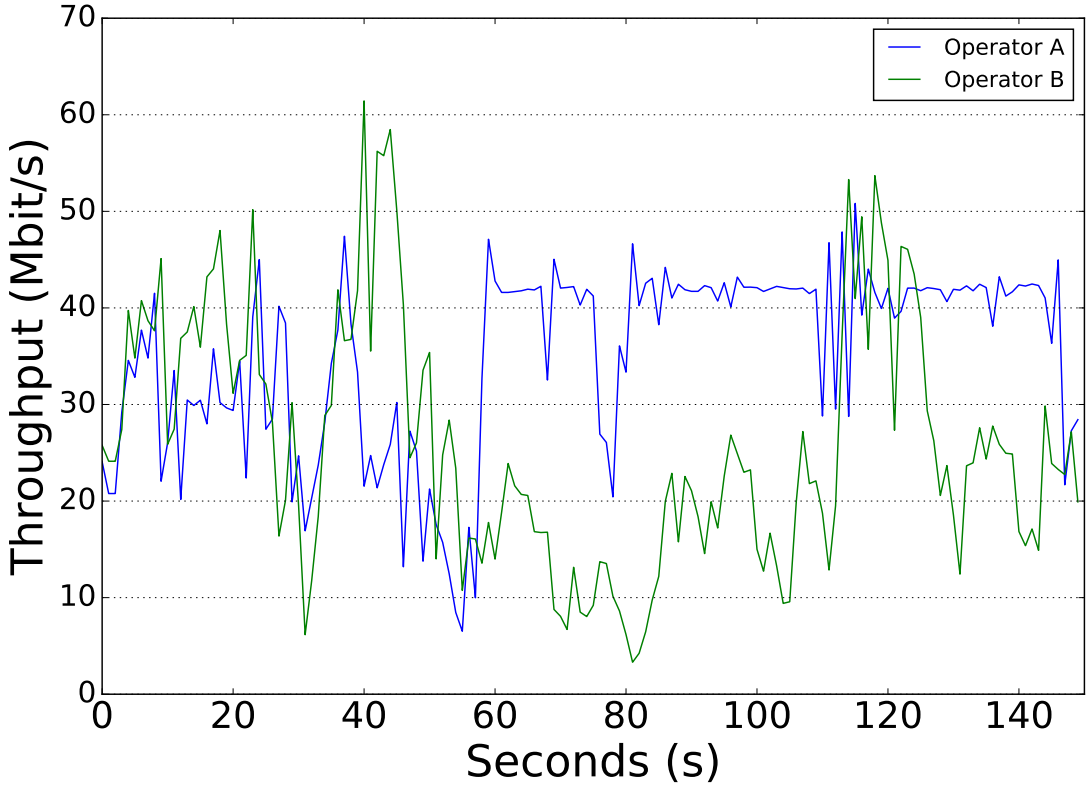


Figure 3.3: Time-series of application throughput for car mobility pattern and two different mobile operators

3.1.3.2 Channel

Measured throughput is a combination of the eNodeB environment (load, scheduler policy), wireless channel characteristics and mobile device receiver capabilities. Additional information about the channel environment other than throughput values can increase accuracy and granularity, paving a way to more accurate prediction. In Figure 3.4 and 3.5, we analyse this relationship and show boxplot of CQI against application throughput for operators A and B respectively. The boxplot shows the range of throughput values for each CQI separately. Overall, we can observe an increasing trend in throughput proportional to CQI. However, the range of throughput values oscillates significantly for each CQI. Furthermore, for operator A, the average throughput of CQI equals 14 is lower than the throughput for CQI 15. A similar observation holds for operator B as well. Finally, this result is strengthened even more with the calculation of Pearson correlation (which depicts linear relationships) between throughput and CQI, yielding a correlation coefficient of 0.6 and 0.38, for operator A and B, respectively. However, this correlation is even lower for other cases; in particular for the static case where the correlation coefficient equals 0.35. While CQI shows correlation with

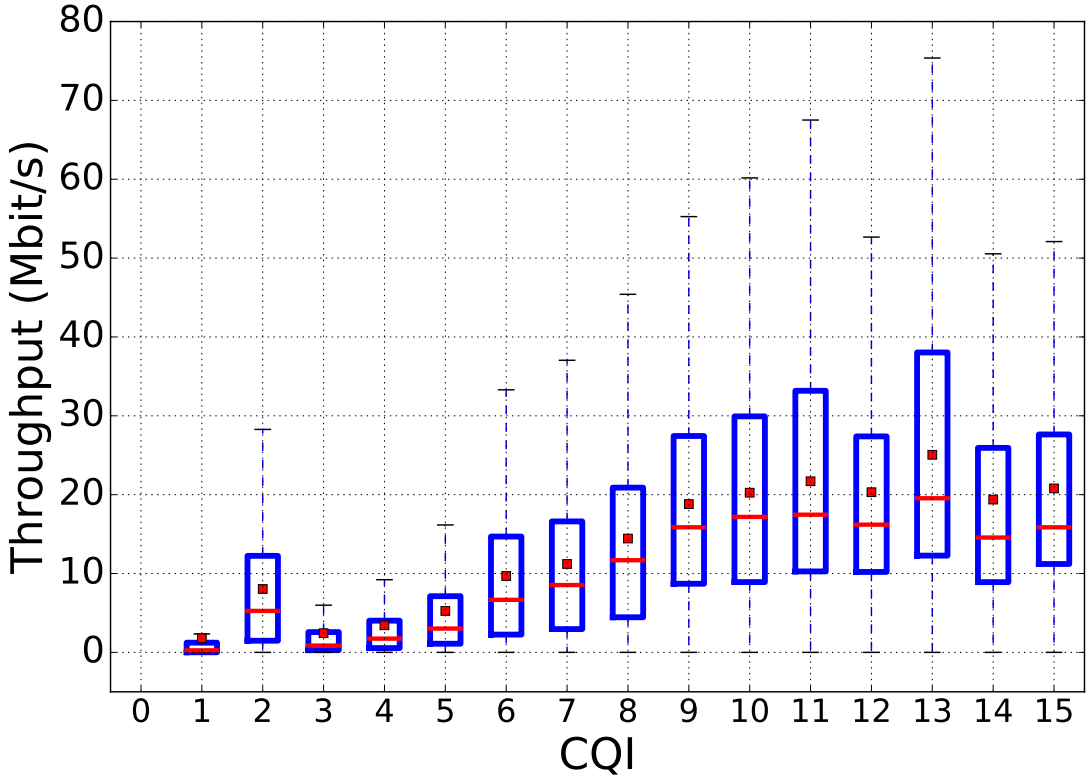


Figure 3.4: Boxplot of CQI vs application throughput (car mobility pattern) for network operator A

throughput, the actual values may be lower than expected. The CQI is calculated on the mobile device (based on wireless channel condition) and represents the maximum rate the device can receive with low error (taking only channel and devices characteristics). However, the actual rate (number of allocated resources blocks per frame) depends on multiple factors and not CQI alone. For example, a user may have lower throughput demand downloading on a lower rate than the available bandwidth. Also, depending on the number of users sharing network bandwidth (users connected to the same base station), eNodeB scheduler may assign lower throughput to the user to accommodate other user's throughput demands. This explains why the correlation coefficient is in the lower region (∼0.6) and not higher than 0.8. This observation is consistent with the results obtained in [YJS+18].

3.1.3.3 Context

The dataset provides additional context information such as the device's GPS positions and velocity. Figure 3.6 shows GPS coordinates of all measurement points

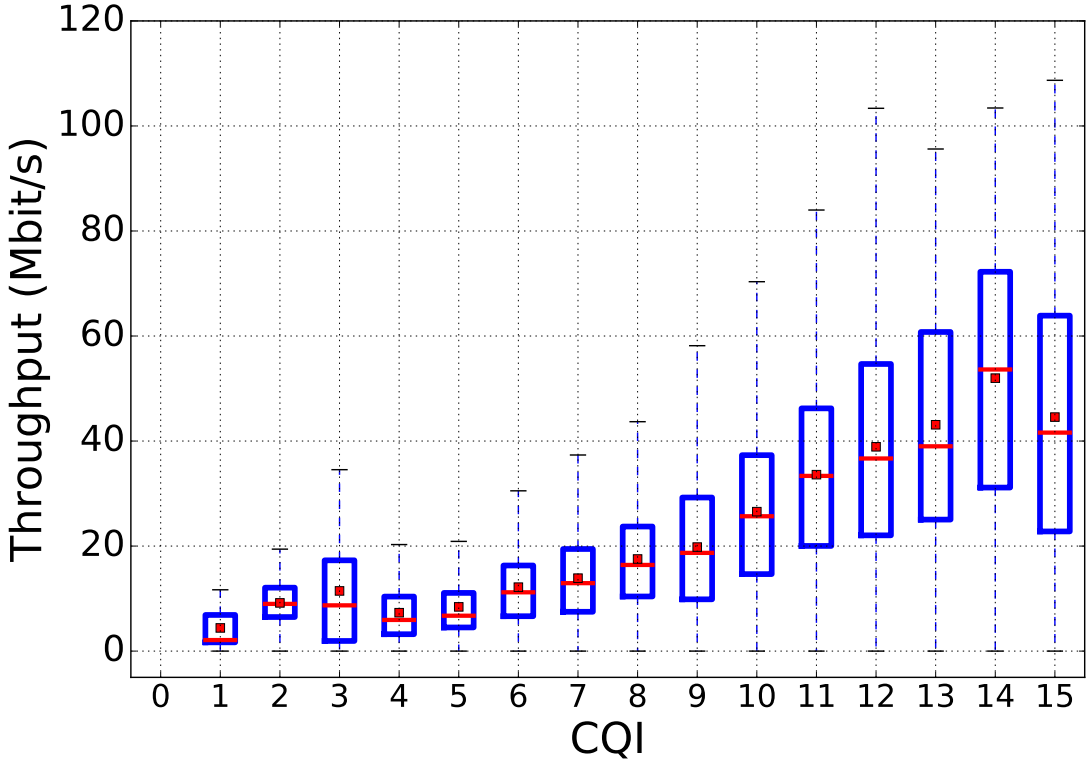


Figure 3.5: Boxplot of CQI vs application throughput (car mobility pattern) for network operator B

from our dataset. We provide estimated GPS coordinates of serving eNodeBs and distance between them using Haversine formula [Bru13].

Additionally, Table 3.4 shows the average and variation range of device velocity across different mobility patterns and network operators. Intuitively, speed increases as we move from static (not shown) to pedestrian and finally train scenario. A similar observation holds for variation range as well. Velocity values are alike for both network operators as the same phones/patterns were used for both operators.

Table 3.4: Average and Variation Range of device velocity (kph) across different mobility patterns and mobile operators

	Mobility Patterns							
Operator	**Pedestrian**		**Bus**		**Car**		**Train**	
	Avg.	*Var. Range*	*Avg.*	*Var. Range*	*Avg.*	*Var. Range*	*Avg.*	*Var. Range*
A	2.4	(0.0, 4.0)	17.2	(0.0, 34.0)	23.7	(0.0, 54.0)	60.6	(0.0, 109.4)
B	1.5	(0.0, 3.0)	10.7	(0.0, 30.0)	35.1	(0.0, 56.0)	53.9	(0.0, 114.0)

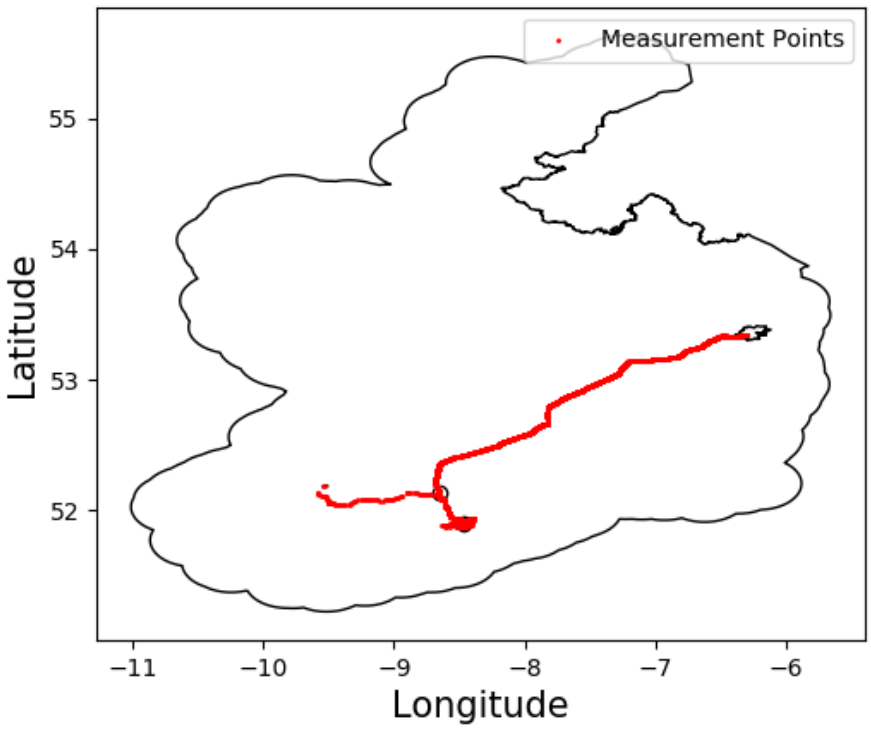


Figure 3.6: GPS coordinate for the all measurement points across Ireland

3.1.3.4 Caveats

This production dataset contains a considerable amount of information. However, there are several limitations. First, the sampling granularity is only one second. This limitation is due to G-NetTrack and the Google channel Application Programming Interface (API). Even with direct access to the API, granularity does not significantly increase [YJS+18]. Second, not all records have all the values. The most prominent example represents RSSI, which isn't logged for every sample. Similarly, geolocation of eNodeB is obtained from the `opencell.org` database. Unfortunately, this database doesn't contain GPS coordinates for all eNodeBs. One approach to deal with missing data is to use one of an imputation method. Several imputations methods exist, from replacing missing values with simple mean or median, or using simple machine learning algorithms like k-nearest neighbours algorithm [BM03], to more sophisticated techniques like soft-impute [MHT10].

Autocorrelation Function (ACF), measures the linear dependence between the current and past values of a variable.

The rationale of ACF is that low values (<0.4) indicate that past values of a metric do not bring many benefits in predicting its current value, either because the past values are too old or because of the intrinsic randomness associated with the metric. High ACF values (>0.8) suggest instead that incorporating such past values is beneficial in predicting the current and future values of the metric.

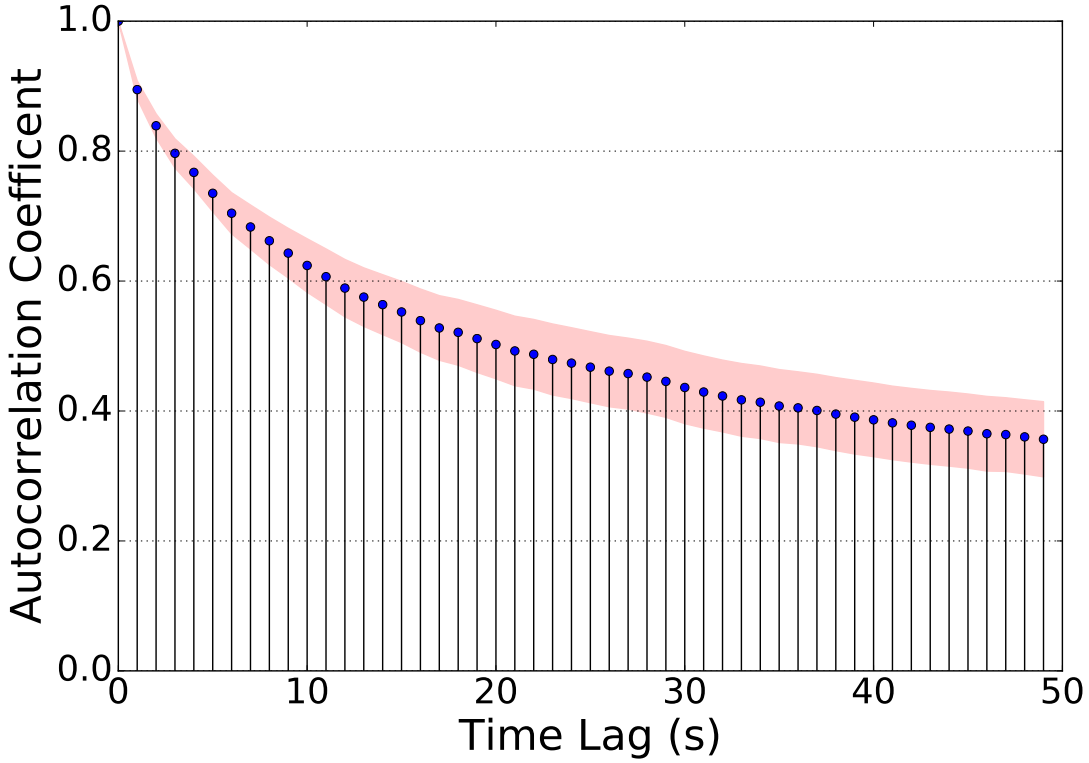


Figure 3.7: The autocorrelation coefficient of throughput (car mobility pattern)

Figure 3.7 shows the average ACF computed across all car mobility pattern traces collected of the application throughput metric. *Time lag* denotes how far in the past do we consider the value of the metric, and we vary it between 0 and 50 seconds. The shaded area represents a standard deviation band. From the figure, it is clear that after the 38-second lag, the autocorrelation coefficient goes below 0.4, indicating no significant correlation after a 38-second delay. However, for other traces, the coefficient value after 20 seconds into the past is low being indistinguishable to noise.

3.2 Testbed frameworks

The following chapters analyse HAS performance in wireless (cellular in particular) and wired environments. Experiments in the wireless environment consider

one device under various mobility patterns represented by the 4G dataset explained in the previous section. To represent a typical mobile user and its environment as realistically as possible, the wireless testbed uses a real mobile device and wireless access point connected to the video server. On the other hand, wired experiments are concerned with multiple heterogeneous clients, representing a typical home environment. In this case, the Mininet emulation environment is used for emulating a large number of competing users. While the testbeds are distinct enough to warrant two separate implementations, some components are shared by both implementations. To minimise repetition, the main characteristics of a testbed for wireless experiments are explained, followed by details about key different components composing the wired-based testbed.

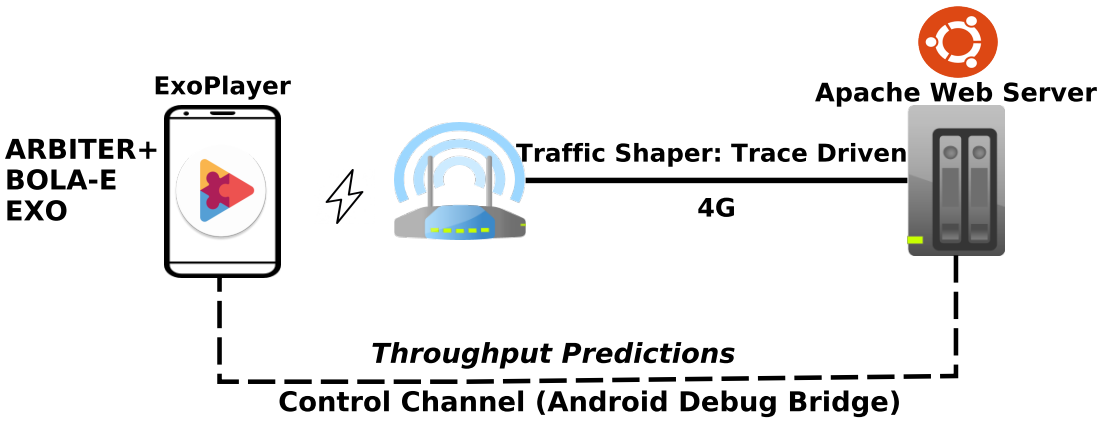


Figure 3.8: Testbed Architecture

3.2.1 Testbed for wireless experiments

Figure 3.8 depicts the testbed architecture which consists of a mobile device (Nexus 6 running Android Operating System (OS) version 7.1.1), a wireless Access Point (AP), and a server (PC running Ubuntu 16.04 equipped with 16GB of RAM and Intel i7 CPU). The mobile device streams video content from the server through the AP. Simultaneously, through Android Debug Bridge (ADB), additional controls and information can be transmitted directly to the device. ADB is a powerful client-server program that allows communication with the Android device. It provides functionalities such as debugging and installing app and access to Unix shell. ADB consists of three components: *client* (sending commands to the device), *daemon* (running on Android device) and *server* (manages communication between client and daemon). ADB allows communicating with

the device either through WiFi or USB [4].

The server also acts as a *traffic shaper* by inserting bandwidth profiles from 4G traces (Section 3.1.2) between itself and the AP. This is achieved using Linux *traffic control* (tc[5]). Modelling link capacity from real traces implicitly reflect the impact of cross-traffic on available throughput. Usage of Mininet and tc-based bandwidth shaping results in emulation-based experiments. Experiments are run in real-time on PC. As a consequence, each experiment can produce different results (e.g. running tc multiple times on the same bandwidth trace). To address this, multiple runs for the same bandwidth trace were performed. However, HAS algorithm operates on a discrete set of bitrates, so changes in bandwidth profile do not have a significant impact. Experiments are run in real-time and repeated for each bandwidth trace. This limits the number of runs. As the balance, ten runs were selected.

The mobile device runs a video player built using ExoPlayer[6]. ExoPlayer is a highly customisable open-source application-level media player for Android. It supports HAS technologies (e.g. Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS)) for video streaming. ExoPlayer is written in java. ExoPlayer supports the DASH standard via a stand-alone library and also provides a default adaptation algorithm, which referred to as EXO in the remainder of the text.

3.2.2 Testbed for wired experiments

In wired experiments, the base assumption is that link capacity is constant due to medium (wire) properties compared to wireless channel characteristics. These experiments quantify scenarios such as broadband home access network with few users or edge part of the network with tens of users competing for network resources. Due to stable throughput capacity, a disturbing factor for application performance comes from cross-traffic generated by other applications. The goal is to have a platform that supports a large number of users generating real traffic from various applications over a bottleneck in real-time. One solution is to use the full hardware approach, representing each user with a physical hardware device (e.g. PC or laptop). However, this is a costly approach limiting the number of competing user. Alternatively, simulation is the most cost-effective solution and

[4] `https://developer.android.com/studio/command-line/adb`
[5] `https://wiki.debian.org/TrafficControl`
[6] `https://exoplayer.dev/`

virtually can support a large number of users. Still, simulation is based on mathematical models for the network, channel, protocols and applications. Finally, network emulation acts as the middle ground, allowing running and assessing the performance of real applications over a virtualised network, and is the approach taken for creating the testbed.

The testbed platform is created using the popular open-source Mininet system [LHM10], which allows emulating large networks in real-time. By employing virtualisation, Mininet can scale experiments at low cost with matching accuracy compared to an actual hardware experiment. All the clients, server and switches are modelled as Linux nodes within the Mininet environment. Mininet uses real Linux code for TCP, queuing mechanisms, etc. thus increasing realism in the evaluation. Experiments are performed on a single laptop with Intel i7 CPU and dedicated GPU (Graphics Processing Unit) coupled together with 16GB RAM and 500GB SSD drive.

Note that using real traces in wireless scenarios implicitly reflect the impact of cross-traffic on available throughput. However, as cross-traffic characteristics are unknown, further qualification of impact, either on video performance by cross-traffic or vice versa, is not possible. On the other hand, full network emulation allows for further analysis of the interaction between video and different types of traffic.

Figure 3.9 depicts a common dumbbell topology for wired experiments. This topology is commonly used in scenarios where multiple clients share network resources [CCPM13, JSZ14, LZG+14, AABD12, HG12, HJM+14, MVSA13]. N clients are sharing a bottleneck link and request their data from a remote HTTP server. Video and web content is stored on the HTTP server. The role of the bottleneck link and network elements is to emulate scenarios with different queue size, queue scheduling techniques, link capacity, and round-trip-time (RTT). Performance metrics for different applications are collected at the client-side.

GPAC is used as the video player. GPAC (ver. 0.5[7]) a well-known open-source video player that supports the HAS technique. GPAC is a full-fledged player that decodes video content in real-time. By default, GPAC uses a HAS algorithm that requests the representation whose rate is just below the throughput of the last downloaded segment.

[7] `https://gpac.wp.mines-telecom.fr/`

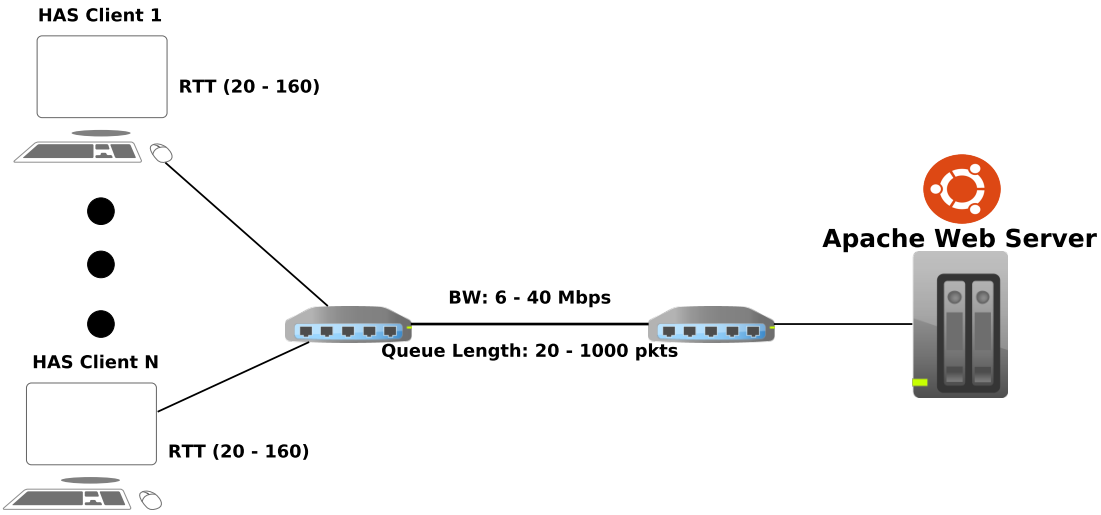


Figure 3.9: Testbed framework architecture for wired experiments

3.2.3 Key features of frameworks

3.2.3.1 Video content

For video content, the High Definition dataset [QZS16] is used. The dataset provides both Advanced Video Coding (AVC) (H.264), and High Efficiency Video Coding (HEVC) (H.265) encoded video content, with 23 clips encoded in ten different rates. The duration of the video clips ranges between 10 and 14 minutes. For segmentation and encoding the dataset, the following tools were used: FFmpeg - encoding original content to lossless YUV format; x264/x265 codec for encoding to AVC/HEVC format; MP4Box for segmentation and multi-profile MPD generation. Furthermore, the video content is generated for five different segment durations (2, 4, 6, 8 and 10 seconds), allowing for a greater variety of experiments. This helps a researcher to better quantify the impact of different segment durations on the underlying adaptive delivery service. Table 3.5 shows the encoding settings used for the dataset (frame rates depend on video clip).

3.2.3.2 Implemented HAS algorithms

Every video player is enhanced with additional adaptation algorithms. Furthermore, after video players finish streaming, a log file is saved. All logging functionalities are additionally added to video players. While players support logging, these logs are mainly used for debugging purposes. For this reason, a new logging library is added to each player, allowing easier calculation of video-related

Table 3.5: Ladder for the average HD encoding rate, resolution and frame rate for the used dataset [QZS16]

#	Bitrate	Resolution	Frame Rate
10	4.3Mbps	1920x872	24, 60
9	3.85Mbps	1920x872	24, 60
8	3Mbps	1280x582	24, 60
7	2.35Mbps	1280x582	24, 60
6	1.75Mbps	720x328	24, 60
5	1.05Mbps	640x292	24, 60
4	750Kbps	512x234	24, 60
3	560Kbps	512x234	24, 60
2	375Kbps	384x174	24, 60
1	235Kbps	320x146	24, 60

metrics. Table 3.6 shows an example of the output from the player. The logs contain values for each segment selection decision. Within this log file, we provide a rich set of various metrics, which can be used for later analysis. In addition to the standard performance metrics (quality rate, buffer level, and stall duration) which are commonly used to generate QoE related metrics (Section 3.2.3.4), such as the average video quality, switching rate (i.e. instability) and video stalls (i.e. total stall duration and number of stalls). Detailed information is provided for each segment (its delivery rate, actual rate, which is a function of segment size and segment duration).

Table 3.6: Sample trace output from modified dashc

Seg_#	Arr_time	Del_Time	Stall_Dur	Rep_Level	Del_Rate	Act_Rate	Byte_Size	Buff_Level
1	109	109	0.000000	232	9070	248	124131	4.000
2	1375	59	0.000000	232	18704	276	138452	8.000
3	3116	533	0.000000	4275	39881	5323	2661696	11.466
4	4621	268	0.000000	4275	47542	3187	1593595	15.198
5	6012	113	0.000000	4275	53917	1524	762041	19.085

The selected HAS algorithms are widely used in the literature and show good performance in various conditions. These algorithms are based on different design requirements, i.e. rate-based (FESTIVE and Conventional), buffer-based (LOGISTIC and BOLA-E) or hybrid (ARBITER+ and BBA-2). Also, they use different bandwidth estimation techniques (e.g. harmonic and exponential moving average) which makes the basis for additional classification on more conservative (e.g. FESTIVE and ELASTIC) and more opportunistic algorithms (e.g. ARBITER+). Note, while frameworks implement a relatively large number of HAS algorithms, not all algorithms are used for evaluation in forthcoming chapters.

The following adaptation algorithms are implemented across video players:

- The BBA-2 algorithm [HJM+14] mainly selects the next segment representation by mapping the buffer level to a target segment size. It also incorporates a throughput-based decision in its "startup" phase while taking into account future (up to 480 seconds) video segment sizes.
- The FESTIVE algorithm [JSZ14] is a rate-based algorithm that uses a harmonic estimator for the network throughput. It employs three components, namely, randomised chunk scheduling (avoiding ON/OFF issues), stateful bitrate selection (the algorithm will switch to the next subsequent higher rate only if it stream at the current rate for a couple of chunks) and a delayed update approach (trade-off between frequently switching between different rates and staying on the same rate for a longer period. This component monitors the number of switches over the last 20 seconds to decide should the player switch to different rate).
- The ARBITER+ algorithm [ZRS18] is a hybrid-based algorithm that uses the EWMA of the last ten chunk rates. Alongside EWMA Arbiter+ employs two additional rate scale factors, to track variation in throughput samples and buffer occupancy. The first factor tracks variation in throughput samples and reduces the estimated rate if bandwidth fluctuation increases. The second one tracks buffer occupancy drain and lowers the rate if the buffer is too low to prevent stalls. However, for higher buffer levels, this factor will increase the bandwidth estimate, resulting in the algorithm being less cautious and requesting a higher quality bitrate. This potentially increases downloading time and thus slows buffer saturation avoiding OFF phase.
- The ELASTIC algorithm [CCPM13] is a hybrid-based algorithm which uses a harmonic average of the recent five segment rates. The harmonic average is a conservative estimate of available throughput (by its nature, harmonic mean is a conservative estimate, AGH inequality [Mer14]). Furthermore, Elastic employs control-theory to combine the throughput estimate and buffer levels when making video rate selection decisions.
- *Conventional* - Conventional [LZG+14] represents a rate-based adaptation algorithm. It makes its choice on the next segment, by using an exponential moving average of past segments delivery rate.
- The BOLA-E algorithm is an extension to BOLA adaptive algo-

rithm [SUS16]. BOLA is a buffer-based algorithm that relies on Lyapunov optimisation to maximise video rate and minimise stall (rebuffering) events. The utility function increases with average bitrate, while the increase in stalls reduces it. However, the algorithm is flexible to allow optimisation of different QoE metrics (by defining different utility functions). BOLA-E [SSS18] introduces a throughput estimate to improve startup, seek and low-latency performance of BOLA. The throughput estimate is an average of the last five chunk delivery rates.

- The LOGISTIC algorithm [SME15] is a buffer-based algorithm that selects the next segment quality by mapping the buffer level to a target segment quality. Logistic uses a log function to map buffer levels to video bitrate.
- The EXO algorithm is a hybrid type of HAS algorithm [8]. For rate estimation, it calculates the sliding median of a specified number of recent chunk rates, where a chunk is taken into measurement only if the size of the chunk is at least $512KB$ or download duration took at least 2 seconds. Intuitively, median (resistant to outliers) also represents a slightly less conservative estimate compared to the harmonic mean. The buffer is used to decide for requesting already downloaded segment with a higher rate. If the buffer level is high enough, the algorithm will try to download the last segment with higher quality. This is one approach in alleviating OFF phase and improving overall quality.

3.2.3.3 Web behavioural traffic models

The web behavioural traffic model is based on previously published and widely used studies regarding user behaviour during a web session and types of web content.

Web clients are based on Firefox that is driven by Selenium, a web browser automation tool[9]. A user typically opens a page, spends some time on it (*dwell time*) before proceeding to the next page [TZS14]. The dwell times depend on the type of the content and attention span of the user [TZS14]. Dwell times are modelled with a Weibull distribution, whose scale parameter λ depends on the type of content, as justified and detailed in [LWD10]. In over 80% of their data, dwell times are in the range between 2 and 70 seconds. Selected web pages

[8] `https://exoplayer.dev/`
[9] `http://www.seleniumhq.org/`

represent the top 250 most visited pages in the following categories [LWD10]: *Science, Travel, Recreation, Computers, Entertainment, Finance, Relationships, Education, Society, and Vehicles*. These categories were taken from [LWD10], for which the authors have provided parameters to generate appropriate dwell times. The HTTrack tool[10] is used for downloading the entire webpages and storing it on a local webserver. The median size for all webpages is 4058 kB, while 1st and 3rd Quartiles are 939.5 and 13007 kB, respectively. Very small page sizes (0.7 kB) represent sites that only have a redirection link (e.g. www.google.com). Because our experiments are controlled, web clients cannot access those links, which is manifested as a *not found page*. Although not frequent, we decided to leave these redirection links to simulate situations when the user open this type of page on the Internet. When this happens, our client will immediately move to the next page as they would in a real-world scenario.

The web user randomly selects a page, persists on it for some random time which depends on the content [LWD10], and then proceeds to request the following page. If a page loading time is greater than 15 seconds, the user abandons the page, selecting the next one from the list [IT14].

The main issue with web browsers such as Firefox are the memory requirements limiting the number of concurrent clients running at the same time. In a similar vein as *dashc* a scalable web behavioural browsing client is needed. Many researchers analysed web content and user behaviour when browsing [BC98, LGC07, HL99]. However, few web behavioural tools are released to the research community. The existing ones (e.g. SURGE [BC97]) are developed over two decades ago and do not represent the current webpage content architecture. To fill the gap, the *SPEED*, Scalable Python wEb bEhavioural moDel, is developed.

The Firefox web browser can be modelled as an on/off process [HL99] as depicted in Figure 3.10.

The "ON" process consists of requesting and loading the webpage after the user moves to the "OFF" phase (reading time of the page). Then, the user requests the next webpage and process repeats. The webpage incorporates the main object which holds the structure of a webpage, including information (links) about additional elements (inline objects) such as images, scripts, and stylesheets. At the transport layer, the client opens a TCP connection to download the main object. After parsing the main object, the client starts parallel TCP connections to

[10] `https://www.httrack.com/`

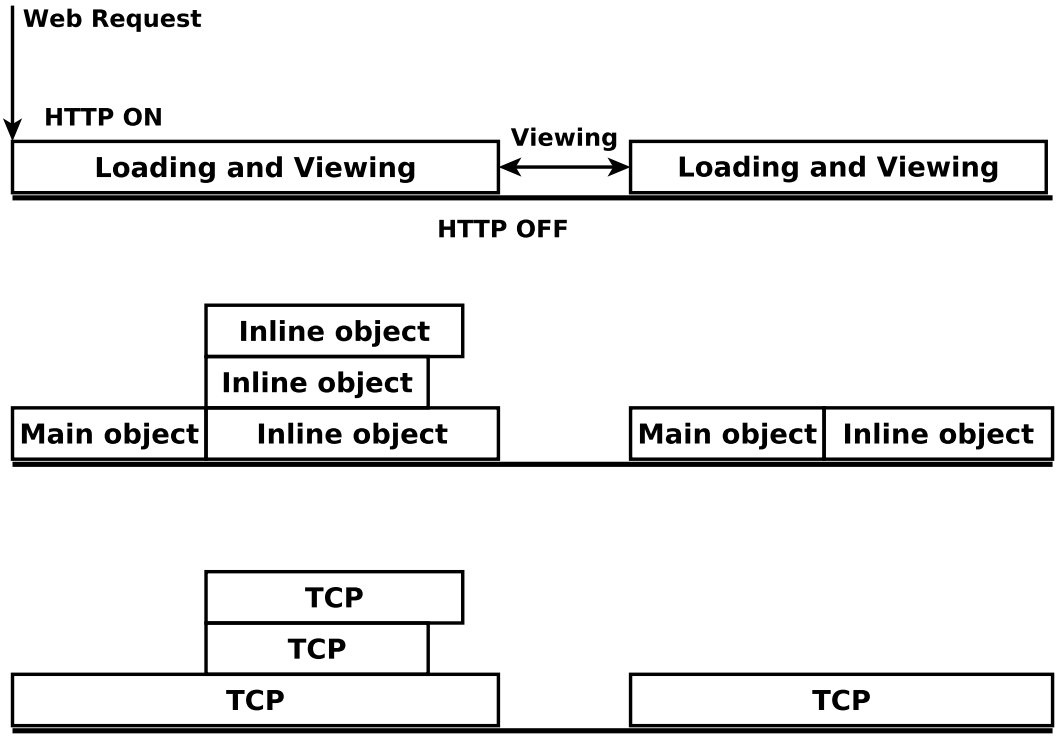


Figure 3.10: Simplified Web Behavioural Model

download the remaining inline objects. By default, Firefox uses up to six parallel connections.

SPEED is based on *aiohttp*[11] asynchronous HTTP client/server library, which allows the opening of multiple parallel TCP connections to a web server. *SPEED* doesn't decode actual objects. This feature demands a custom creation of webpages and objects. Pries et al. [PMT12] performed large-scale measurements, including one million webpages. From these measurements, the authors provide statistical data about webpages, including details such as size and the number of main and inline objects. These stats are modelled using well known statistical distributions that provide the best fit to empirical measurements. Table 3.7 summarises the model parameters used for the synthetic generation of webpages. The remaining parameters (reading time) are modelled the same as for the Firefox-based web behavioural model.

[11] `https://aiohttp.readthedocs.io/en/stable/`

Table 3.7: Webpage content parameters [PMT12]

Parameter	Mean	Median	Max	Standard Deviation	Best Fit
Main object size	31,561 Byte	19,471 Byte	8*MB*	49,219 Byte	Weibull (28242.8,0.814944)
Num. of main objects	2.19	1	212	2.63	Lognormal μ= 0.473844, σ = 0.688471
Inline object size	23,915 Byte	10,284 Byte	8*MB*	128,079 Byte	Lognormal μ = 9.17979, σ = 1.24646
Num. of inline objects	31.93	22	1920	37.65	Exponential μ = 31.9291

3.2.3.4 Performance metrics

To evaluate the performance of HAS algorithms, we analyse standardised QoE metrics, such as average video bitrate, switching behaviour (e.g. stability), stall frequency and duration.

The bandwidth utilisation metric captures the total amount of bandwidth that all video clients are using. Even if they have equal shares of available bandwidth (video clients streaming with the same bitrate), there is a chance that they do not utilise the bandwidth efficiently. The average bandwidth utilisation represents the percentage of bandwidth used over the session lifetime and is estimated as the ratio of the transmitted data to the maximum amount of data that could have been sent during the session activity.

The player instability metric (i) captures the frequency of performing quality switches and is evaluated as [JSZ14, AABD12]:

$$i = \frac{\sum_{d=0}^{k-1} \mid b_{x,t-d} - b_{x,t-d-1} \mid \cdot w(d)}{\sum_{d=1}^{k} b_{x,t-d} \cdot w(d)} \tag{3.1}$$

where k is usually set to 20 seconds, $b_{x,t}$ is the bitrate for client x at time t and $w(d) = k - d$ is a weight function for adding a linear penalty for recent switches. Clients performing more quality switches have higher instability values.

Stall performance is one of the most critical factors that can influence the Quality of Experience [SES+15]. A stall is experienced when the client is forced to pause the video because it has insufficient data in its buffer. Stall duration is expressed in second in all the results in the remaining chapters.

However, to compare algorithms performance, these metrics cannot be studied independently. For example, an algorithm achieving the highest average throughput but frequent stalls is inferior to a more cautious algorithm with no stalls, as it provides a better end-user experience. Capturing subjective user experience and mapping it to objective score is a challenging task. To alleviate this issue, we select two QoE models developed for HAS. These models blend individual QoE metrics to compute a score representing user QoE. Both models are derived from

subjective testing of users grading video clips with various induced impairments. The first model (*Yao QoE*) was derived from data collected in a lab environment [LDU+15]. This model shows a high accuracy for up to five minutes long video sessions. Such limitation does not apply to the second model, which relies on data crowdsourced from users watching videos posted on a website [DDR13]. However, in their evaluation, the authors did not consider stall events. As a result, we select an enhanced model [PFC+15], which extends the preceding model with stall information (*Clay QoE*). The following equations summarise the scores derived from these QoE models:

$$Clay_QoE_{score} = \nu \times QoE_{max} - (\kappa_{TQ} \times I_{TQ} + \kappa_{VQ} \times I_{VQ}) \tag{3.2}$$

$$\begin{aligned} Yao_QoE_{score} = \nu \times QoE_{max} - (\kappa_{TQ} \times I_{TQ} + \kappa_{VQ} \times I_{VQ}) + \\ + \Upsilon(I_{TQ}, I_{VQ}) \end{aligned} \tag{3.3}$$

Where I_{TQ}, and I_{VQ}, represent temporal and visual quality impairment factors, respectively. Similar, κ_{TQ} and κ_{VQ} represent their respective weights. Temporal quality impairments refer to degradation due to initial delay and stall events (stall number and stall duration). Analogously, visual quality impairments take into account average rate and switching behaviour. QoE_{max} indicates the maximum value (score) of QoE or growth factor depending on QoE model. Similar to impairment weights, ν is a weight for the QoE_{max} score. Finally, $\Upsilon(I_{TQ}, I_{VQ})$ represents a cross-effect function of impairment factors occurring simultaneously. When multiple impairments happen, their cumulative subjective effect is not simply the sum of individual impairment [LDU+15]. Function Υ compensates for this effect.

The Yao QoE score starts at 100 and is reduced by impairments, compensated by Υ function of the stall, switching and initial delay impairments. The Clay QoE initial score is based on average rate, reduced by impairments capturing stall and switching impairment. Clay QoE doesn't include cross-effect compensation function Υ.

Analysing both QoE models across different traces, the following observations are made: Both models perceive stall impairments similarly with high correla-

tion (0.9); Models calculate switching impairment differently (Clay uses standard deviation between rates, while Yao relies on the difference in Video Quality Metric (VQM) between chunks); Unlike Clay, Yao uses a cross-effect compensation function which limits the negative impact of multiple impairments; While Clay QoE is calculated over the entire session, Yao QoE calculation is split into 1-min windows, for which a QoE score is calculated separately. Total QoE equals an arithmetic average of five windows.

Because of the observations mentioned above, when applicable, we use the geometric mean of the two models to represent an overall QoE score. We chose geometric mean instead of the arithmetic mean because model scores are on a different scale. Next, we outline the main QoE metrics and their characteristics.

One of the primary imperatives for the networks with finite available resources (e.g. Internet) is sharing resources equally among competing users, together with high resource utilisation. However, quantifying this proportion through a performance metric is challenging. Usually, some Quality of Service (QoS) metric would be used as a basis for representing this fair share among users (e.g. for video streaming applications, average representation rate for each user is often used to depict this ratio). Nevertheless, a fair QoS allocation does not necessarily translate to fair QoE allocation [Bri07, MFA15].

Therefore, we opt for the preceding QoE model instead of objective metrics introduced earlier. Jain fairness index was one of the most popular fairness metrics used for depicting how equitably resources had been shared among multiple users [JCH98]. However, in their recent work, Hoffeld et al. [HSKHV17, HSKHV18] point out limitations of the Jain index. In summary, the following properties are not satisfied with the Jain index:

1. **Scale and metric independent** - This property means that any linear transformation of underlying QoE metric should not change the resulting fairness index.
2. **Intuitive** - Highest index value should be equal to an entirely fair system, while the lowest index value means a minimum fair system
3. **Deviation symmetric** - index should not depend on sign and value of deviation from the mean value
4. **QoE level independent**

As a result, the authors propose a new QoE fairness metric, F-Index defined as:

$$F = 1 - \frac{2\sigma}{H - L} \tag{3.4}$$

where σ represents a standard deviation of QoE values assigned to each user, while H and L are upper and lower bound of QoE metric, respectively.

For web clients, we capture Page Loading Time (PLT) as a key performance metric. Additionally, we introduce Web QoE based on PLT as defined in [ERHS12]:

$$Q(t) = -a \times log(PLT) + b \tag{3.5}$$

with $a = 1.5$ and $b = 4.25$.

PLT is a more intuitive metric than web QoE. Also, the web QoE model is a function of PLT only. However, for the experiments performed in Chapters 6 and 7, multiple metrics relative to some value are shown in the same figure. For this type of plot, depicting web QoE is more intuitive than PLT (higher values for web QoE indicate better performance while in the case with PLT opposite is true). Still, both metrics are shown in the following chapters.

3.3 Conclusion

This chapter presented tools used for performance evaluation in HAS-based systems and represents the foundation for the following chapters, which uses the presented tools for analysis and proposal of the novel schemes to improve video QoE.

In particular, in addition to presented tools, the chapter made two contributions:

- The 4G trace dataset, with low bandwidth throughput sampling granularity, and client-side cellular channel and context information, from a diverse set of routes across two mobile operators (production) and a large range of clients in a multi-cell cluster (synthetic). The throughput values dataset permit detailed analysis with respect to oscillation in the transmission medium, while the channel and context metrics of the dataset far exceed the original goal of the dataset concerning HAS evaluation for throughput prediction presented in following chapters.
- Flexible, scalable, and customizable real-time framework for testing HAS

algorithms in wireless and wired environments.

Chapter 4

Design Issues with Throughput Prediction for HAS algorithms

This chapter explores integrating throughput prediction in three HTTP adaptive streaming (HAS) adaptation algorithms and quantifies its impact on overall user Quality of Experience (QoE).

The reason for the potential benefits of throughput prediction for HAS adaptation algorithms comes from HAS design itself. The main building component that goes into the decision-making process for the next chunk quality of rate-based algorithms is a throughput estimate from a bandwidth estimator. This single value represents a starting point for algorithm logic, an indicator of how much throughput is available. In other words, this value can be seen as throughput prediction. While both terms can be used in this context, there is a distinct difference. Estimation refers to estimate future values based on by smoothing history values over some predefined window. However, prediction in this context involves trying to predict (infer) actual future value. For the "pure" buffer-based algorithms throughput prediction wouldn't have any impact as algorithm logic solely relies on buffer levels. However, most of the state-of-the-art algorithms use both information to make a more informed decision.

This step is natural due to the modular design of HAS algorithms, that facilitates making changes to the adaptation logic, and would allow existing clients to take advantage of predictions when they are available. Also, neither of previous studies [ZEG+15, MTAA+16] considered applying prediction to existing HAS approaches. These studies propose prediction-based HAS algorithms and demonstrate that they can improve the streaming performance in comparison to typical

HAS algorithms.

More specifically, this chapter focuses on addressing the following research questions:

1. *How the predicted throughput should be integrated in the adaptation algorithms?*

   There are two choices, feeding prediction values directly to adaptation logic or through bandwidth estimator and perform additional smoothing.

2. *How the predicted throughput horizon may influence QoE?*

   Prediction horizon is simply how far in future prediction value is referring to.

3. *How inaccurate prediction may impact the streaming performance and user QoE?*

   In practice, throughput predictions will be contaminated with the error. However, the effect of error level impact in not apparent at first, justifying the further analysis

These questions are investigated in a real experimental testbed explained in Section 3.2.1 that uses collected real 4G cellular traces (Section 3.1) and real video content. As a result of these experiments, the following observations are made:

- Assisting HAS clients with accurate predictions improve users QoE by up to 40% for all tested algorithms
- Longer prediction horizon helps in eliminating stall events and improving overall switching behaviour
- Even with the 30% induced average prediction error, prediction assisted HAS algorithms can achieve 14% improvement in user QoE

4.1 Analysis of traditional bandwidth estimators used in HAS players

By its nature, quality adaptation algorithms have a modular design. The *bandwidth estimation module* captures the network state, while the *application monitoring module* captures the video player state by monitoring playback buffer and

streamed video quality. Finally, the *bitrate adaptation module* combines information from previous modules to decide the quality of the chunks to be requested.

For the evaluation, three algorithms: ARBITER+, ELASTIC and EXO (see Section 3.2.3.2 for details about each algorithm) are selected. Selected algorithms use information from both bandwidth estimators, as well as from buffer occupancy when deciding on the rate of the next chunk. As bandwidth estimators, ARBITER+, ELASTIC, and EXO use Exponential Weighted Moving Average (EWMA), Harmonic and Median, respectively.

Figure 4.1 shows a time-series of instantaneous throughput (blue line) for a highly-variable trace (see Section 4.3.1) taken from previously collected 4G dataset (Section 3.1). In addition to actual throughput, throughput estimates are plotted from the bandwidth estimators used by each algorithm, i.e. ARBITER+, EXO, and ELASTIC (red, green, and yellow line, respectively).

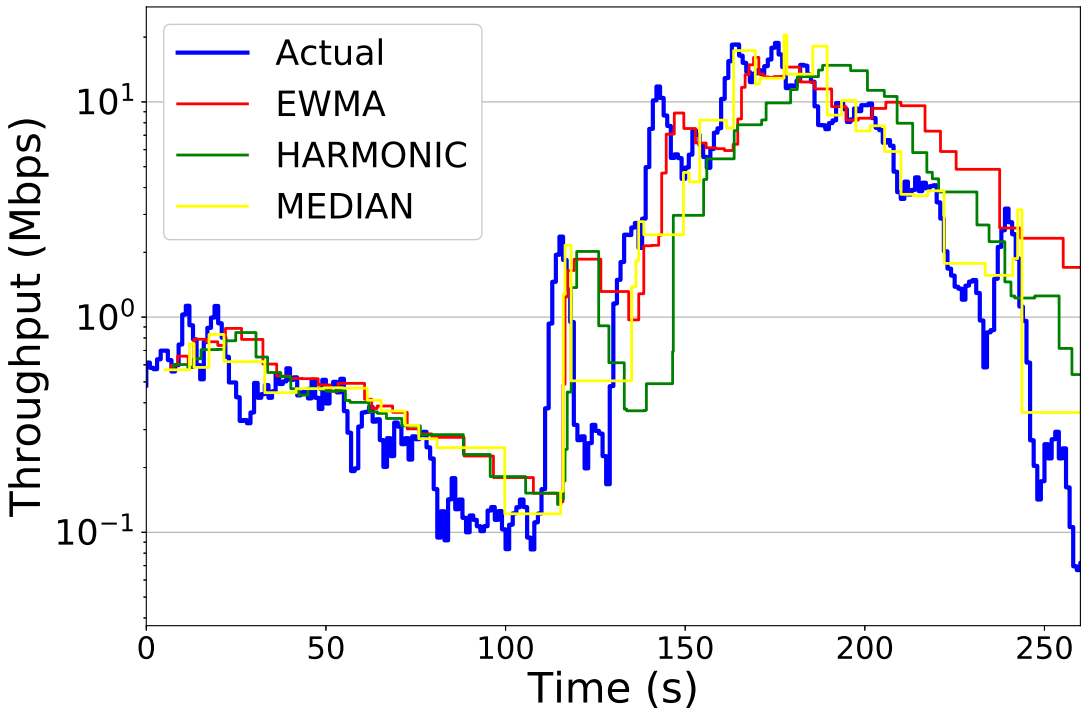


Figure 4.1: Time-series of instantaneous throughput measured every second

EWMA shows the highest average throughput among all bandwidth estimators. Compared to the actual throughput, EWMA tends to overestimate throughput (58% of the times) while harmonic and median show neutral bias, and overall lower values compared to EWMA. Trend-wise, EWMA follows actual throughput closely with resistance to small variations due to smoothing effect. Instead, harmonic reacts to throughput changes slower than both EWMA and median.

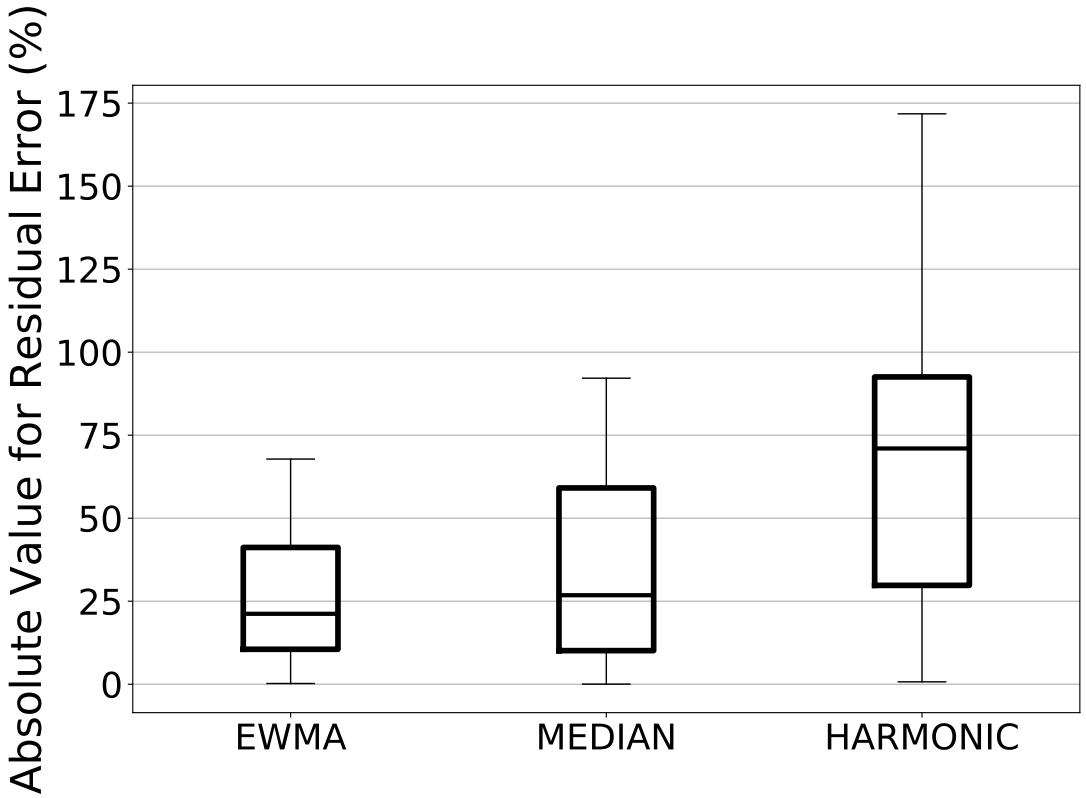


Figure 4.2: Boxplot of residual error for different bandwidth estimators

To further showcase the limitation of current throughput estimation mechanisms, we analyse the absolute value of residual error between throughput estimates and the actual download rate of a chunk. Figure 4.2 shows boxplots of the residual error for EWMA, median, and harmonic, measured while experimenting with the selected 4G trace. Figure 4.2 shows that EWMA achieves, overall, the lowest residual error while median and harmonic perform comparable. Nevertheless, EWMA's average residual error is still 50% (green dot).

The selected trace is characterised with frequent throughput oscillation causing high estimation error of analysed bandwidth estimators. For traces with low variation, the performance of bandwidth estimation would improve, but as shown in Section 4.3.2.2, accurate throughput prediction can still significantly improve user QoE.

From the above analysis, it is intuitive to ask what will happen with the performance of HAS algorithm if bandwidth estimation accuracy improves.

4.2 Integrating throughput predictions with HAS algorithm

Next research challenge is how to feed the prediction value to the algorithm logic. In previous studies [ZEG+15, MTAA+16] authors design specific "prediction-aware" adaptation logic assuming that accurate value is available. However, to only quantify the impact of accurate predictions on HAS performance, design logic of existing algorithms should be intact. Intuitively, this constraint leads to only two ways of feeding predictions to existing algorithms.

The predicted throughput can be integrated into the adaptation logic in two different ways. First, the predicted throughput may replace the entire throughput *estimation* in the algorithm (*E-type*). Alternatively, the predicted throughput may be used to replace the estimated throughput *samples* (*S-type*). Figure 4.3 shows both *E-type* and *S-type* approaches.

The following research question is, what throughput prediction represents regarding the future? When the chunk is downloaded, HAS algorithms decide for the bitrate quality of the next chunk. Therefore, the prediction value should reflect only the near future. In steady-state, this future is equal to chunk duration. This way, buffer drain rate (i.e. playback) and arrival rate of a chunk are equal. However, as throughput varies over time, the algorithm would try to match the arrival rate. Note that bitrate quality of chunk is average bitrate over all chunk sizes divided by chunk duration. Matching arrival rate for every chunk would lead to frequent switching between different bitrates thus lowering user QoE. This leads to another role of bandwidth estimator. It does not only estimate throughput for the next chunk, but it also smooths the estimate value so that switching is minimised.

Comparing *S-type* and *E-type* predictions, *E-type* prediction would result in more switches compared to *S-type* prediction, especially if the prediction horizon is relatively small. Prediction horizon is defined as averaged predicted value over next x seconds into the future. Also, while longer horizon improves switching performance, stall performance can decline because longer horizon can hide sudden drops in throughput (due to averaging).

Another factor that affects performance is the bandwidth estimator. Harmonic and median estimators are by design more conservative compared to EWMA estimator. In this case, the *E-type* approach could have a more positive impact than

S-type as the average quality would increase and potentially outweigh switching instability.

The prediction engine is responsible for obtaining throughput predictions. This prediction can come from the network [KMC17] or from the device itself (by leveraging radio metrics and machine learning techniques, see Chapter 5). For each of these approaches, the impact of the throughput horizon on the performance is further investigated.

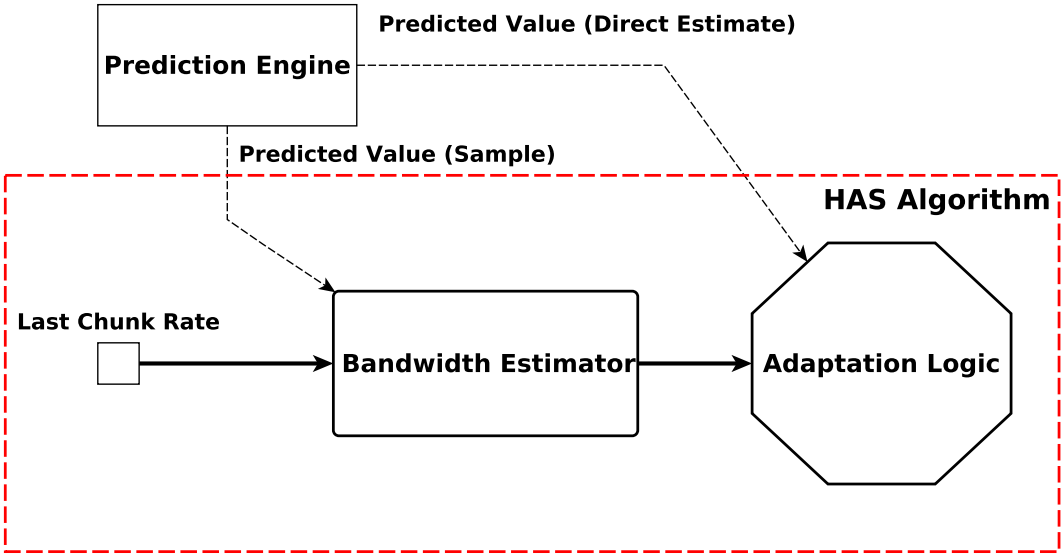


Figure 4.3: Two approaches in feeding prediction to HAS algorithm

4.3 Evaluation

For evaluation, a testbed for wireless experiments is used, described in Section 3.2.1. The wireless environment is more challenging for HAS players compared to wired (on average throughput oscillations in a wired environment are significantly less frequent compared to wireless). All the prospective evaluation is performed on the wireless channel, more specifically in 4G networks. For bandwidth traces, the collected 4G dataset described in Section 3.1 is used. As previously mentioned in Section 4.1, three HAS algorithms are analysed: ARBITER+, ELASTIC, and EXO. The control channel (Android Debug Bridge (ADB)) is used to feed ideal predictions back to the device. Video sessions are 5-minutes long with every scenario repeated ten times and results averaged. Finally, Table 4.1 summarises QoE metrics and its notation.

Table 4.1: QoE Metrics Notation

Metric	Summary
r_{avg}	Average bitrate (Kbps)
i_{avg}	Average instability (i), Section 3.2.3.4
s_{num}	Average number of stalls
s_{dur}	Average stall duration (s)
QoE	Geometric mean of Clay and Yao QoE, Section 3.2.3.4

4.3.1 Trace classification

Based on the results in Mangla et. al [MTAA+16], the standard deviation of throughput is considered within a trace as a discriminant of (potentially) *good* or *bad* input traces. The hypothesis is that prediction will be most effective in the presence of traces with high throughput standard deviation (in particular when the standard deviation is higher than the highest representation rate resulting in bandwidth fluctuations spanning across all available rates). From the dataset, we select highly variable traces with a standard deviation in the range $[4.2, 6.3]$Mbps, and low variable traces with a standard deviation in the range of $[0.6, 1.2]$Mbps are selected. We further filter out traces with very high average throughput (6Mbps), i.e. average throughput larger than the highest video quality (4.3Mbps). The rationale here is to avoid testing scenarios where all algorithms converge to the highest quality level regardless of throughput variation (as fluctuations will get averaged out by the throughput estimator). Out of 130 traces, 26 traces satisfied the latter constraint. As expected, the majority of high-variable traces are collected in the highly mobile environment (car), while low-variable traces where collected while devices were static or moving with low velocity (pedestrian).

4.3.2 Accurate predictions

In the following, the impact of integrating error-free throughput predictions with different horizons on video QoE is explored. Based on the discussion at the beginning of this section, different prediction horizons are evaluated as longer horizons will produce more smoothed values leading to better switching performance (less number of switches) but could potentially decrease average bitrate quality and have deteriorated impact on stall performance.

As a design choice and with no loss of generality, three prediction horizons as multiples of chunk duration, i.e. 4, 8, and 12 seconds are considered. In the

following plots, prediction horizons are associated with the *red*, *blue*, and *green* colour, respectively. As a notation mark, we use S to denote usage of prediction as a sample (*S-type*) and E for prediction as an estimate (*E-type*). We further visually separate sample and estimate results using a diagonal pattern. Finally, note that each metric is normalised with respect to the original algorithm performance (referred to as *no-prediction* setup), whose actual metric values are offered in the figure.

4.3.2.1 Traces with high variability

Results obtained when considering *traces with high variability*, i.e. traces characterised by a high value of throughput standard deviation are analysed first. Figure 4.4, 4.5, and 4.6 show that integrating prediction noticeably improves the QoE metrics of all adaptation algorithms. Specifically, prediction enables all algorithms to reduce/eliminate stalls. Additionally, prediction enables the algorithms to reduce the switching instability. In particular, average instability can be reduced by 12%-37%. Furthermore, improving the accuracy of bandwidth estimation enables the algorithm to enhance their selected chunk quality. For example, integrating prediction fixes throughput underestimation with ELASTIC and EXO leading to a higher chunk quality. On the other hand, integrating the prediction with ARBITER+ results in a lower average quality bitrate. While EWMA tracks changes in throughput reasonably well, most of the time it overestimates available throughput, causing a higher average quality bitrate (see Figure 4.1 for the illustration). All these improvements add up thus boosting the overall user QoE by 23%-55%.

Increasing horizon duration has a positive impact on stall performance and switching behaviour. Extending the horizon results in averaging over a longer period and thus reducing variability between subsequent prediction values. This leads to improved switching performance. Longer horizon enables a client to promptly reduce quality and avoid stalls when the throughput drops for a relatively long time that can deplete the buffer. For all algorithms, 3-chunk horizon shows the highest gain. Longer horizon improves performance even further. However, after the horizon value gets too high (although not shown, this value for an analysed subset of traces is 28-second horizon), overall QoE performance starts to deteriorate as algorithms are unable to avoid stalls.

Results illustrate that the favoured prediction integration approach varies among

different HAS algorithms. Results show that ARBITER+ favours *S-type* integration while ELASTIC and EXO favour *E-type* integration. This is attributed to the nature of EWMA that features a memory element in the throughput estimation. This feature introduces temporal correlation to subsequent estimates leading to improving both stability and quality rate. By passing the throughput as a direct estimate, these benefits are invalidated leading to a degraded streaming performance. On the contrary, conservative estimators (e.g. harmonic mean and median) tend to suppress temporal improvement in network conditions. Note that the median would overlook high throughput samples until they dominate and harmonic mean would deem them outliers. By supplying the throughput directly to the adaptation logic, the adaptation logic would have a better vision of the underlying network changes. However, such benefits become more noticeable with larger prediction horizons.

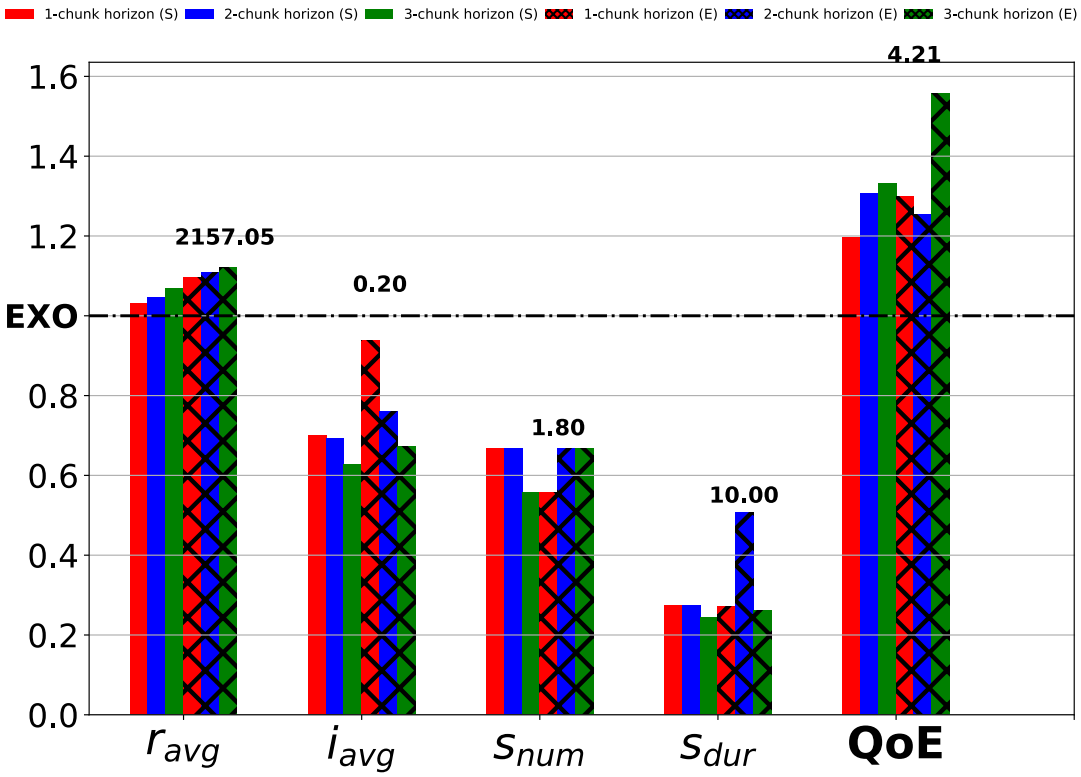


Figure 4.4: EXO: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above bars representing the value of each metric for the no-prediction case*)

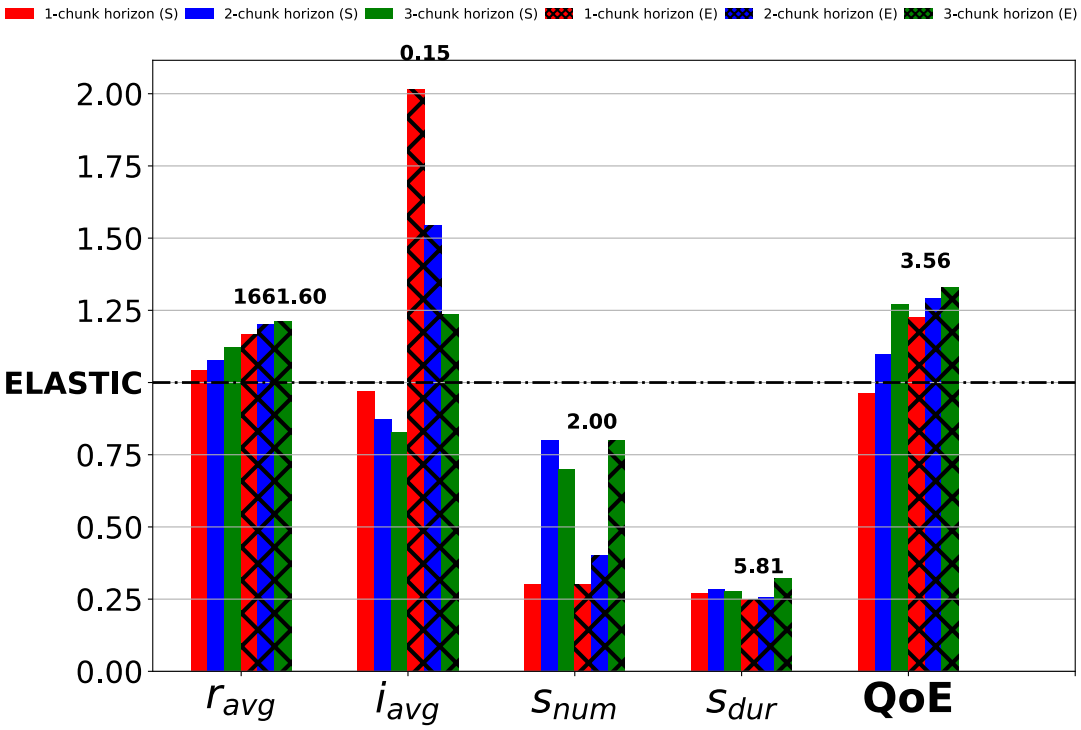


Figure 4.5: ELASTIC: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*)

4.3.2.2 Traces with low variability

Next we focus on results obtained when considering *traces with low variability*, i.e. traces characterised by a low value of throughput standard deviation. Figure 4.7, 4.8, and 4.9 show that HAS algorithms achieve stall-free sessions due to relatively low variation in available throughput. HAS algorithms can counter small variance in bandwidth throughput, as averaging smooths out variations. However, even with the absence of stalls, integrating prediction helps in improving QoE metrics across different algorithms. In particular, prediction improves average instability by 5%-38%. Average representation rate shows a similar trend for HAS algorithms as for the high-variable case. Accurate bandwidth estimates correct throughput underestimation with ELASTIC and EXO, while reducing overestimation for ARBITER+. As a result, overall user QoE improves by 7%-11%.

Similar to the previous scenario, increasing horizon improves switching stability. Overall, the 3-chunk horizon gives the best performance across all algorithms.

Our findings confirm observation from the previous section, with ARBITER+

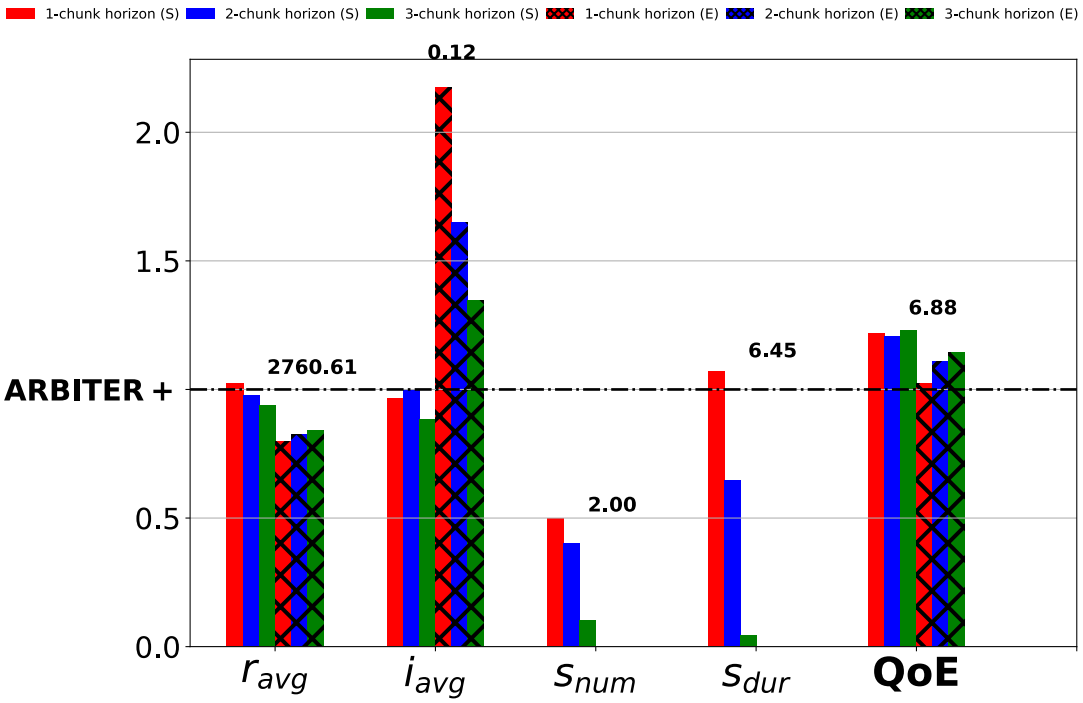


Figure 4.6: ARBITER+: Performance evaluation of QoE metrics for high-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*)

algorithm showing the highest boost with *S-type* prediction, while ELASTIC and EXO prefer *E-type* prediction.

4.3.3 Inaccurate predictions

In practice, obtaining an ideal prediction is unattainable. To evaluate the impact of prediction errors, errors are induced to throughput values for different horizons. Similar analysis has been carried out in [MTAA+16], where the authors add errors to their prediction values. However, their approach is not applicable to existing HAS algorithms as they assumed having multiple disjoint prediction values for a future horizon, with error increasing as the horizon increases (e.g. for 64-second horizon average prediction value is provided for every 4 seconds). On the other hand, having one prediction value for the next x seconds is considered. The error-induced predicted throughput sample $R^E_{Hk_i}$ is the sum of ideal value and error, modelled as:

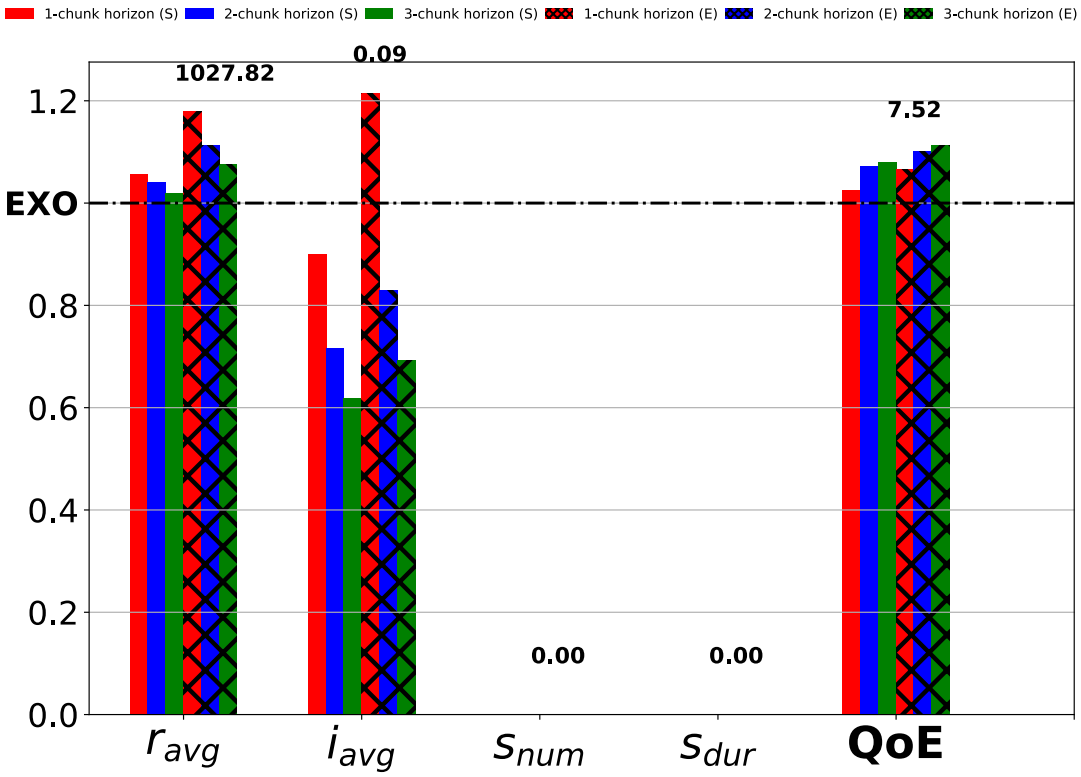


Figure 4.7: EXO: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*)

$$R^{E}_{Hk_i} = R_{Hk_i} + R_{Hk_i} \times N(0, \sigma^2) \tag{4.1}$$

where R_{Hk_i} is a sample i of ideal average throughput over next k seconds (horizon k), $N(0, \sigma^2)$ represents added error term represented as Gaussian (normal) distribution with zero mean and σ^2 variance.

The absolute value of residual error (ARE) is the difference between the actual predicted value and the error-induced values. As a result, mean ARE is mean absolute value of the product of error term and throughput sample ($|E(R_{Hk} \times N(0, \sigma^2)|$). Different values of σ^2 (5, 10, 20, 30) are used to induce an ARE of (5%, 10%, 20%, 30%), respectively.

The impact of adding errors to prediction values is analysed. Experiments are conducted for the traces with high variability only. 3-chunk horizon is used as it shows the highest improvement across all HAS algorithms. ARBITER+ incorporates *S-type* prediction, while direct estimate is used for ELASTIC and EXO.

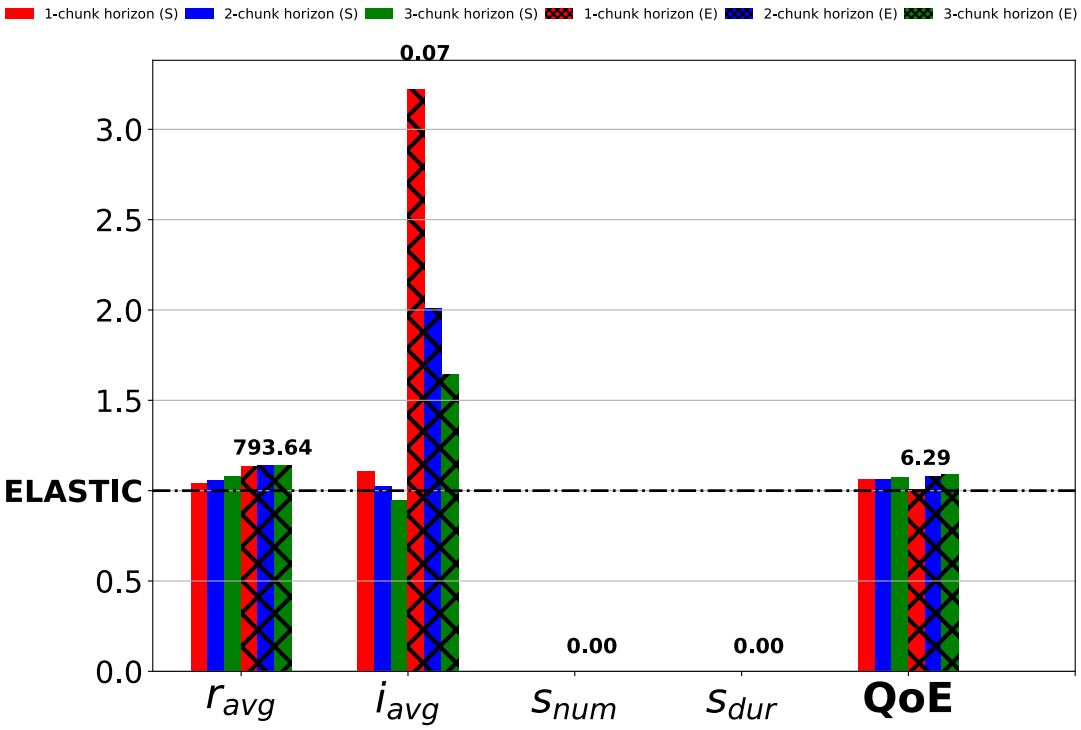


Figure 4.8: ELASTIC: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*)

Figure. 4.10, 4.11, and 4.12 show performance across different QoE metrics and algorithms. Similar to previous sections, normalised values are shown against performance of each algorithm without prediction as the reference point (black dotted line). We use red colour to mark ideal prediction, while predictions with 5%, 10%, 20%, and 30% are coloured blue, green, yellow and gray, respectively. There are no significant differences in average rate quality across different error levels for all HAS algorithms. In particular, even with the 30% induced prediction errors, the difference to (ideal) rate quality is less than 2% for ARBITER+ and EXO, and less than 6% for ELASTIC. On the other hand, switching behaviour and stall performance worsen with the error increase. On average, average instability increases by 60%. This results in higher instability than in the *no-prediction* case when the error reaches 30%. Intuitively, overall user QoE drops with the increase in error levels. However, 5% prediction error lowers QoE by 20% and 4% compared to the ideal prediction for EXO and ARBITER+, respectively, while the loss in QoE is negligible with ELASTIC. Overall, this limits prediction impact. This result is intuitive, as ELASTIC and ARBITER+ employ additional functions limiting bandwidth estimation effects (e.g. ELASTIC uses more cau-

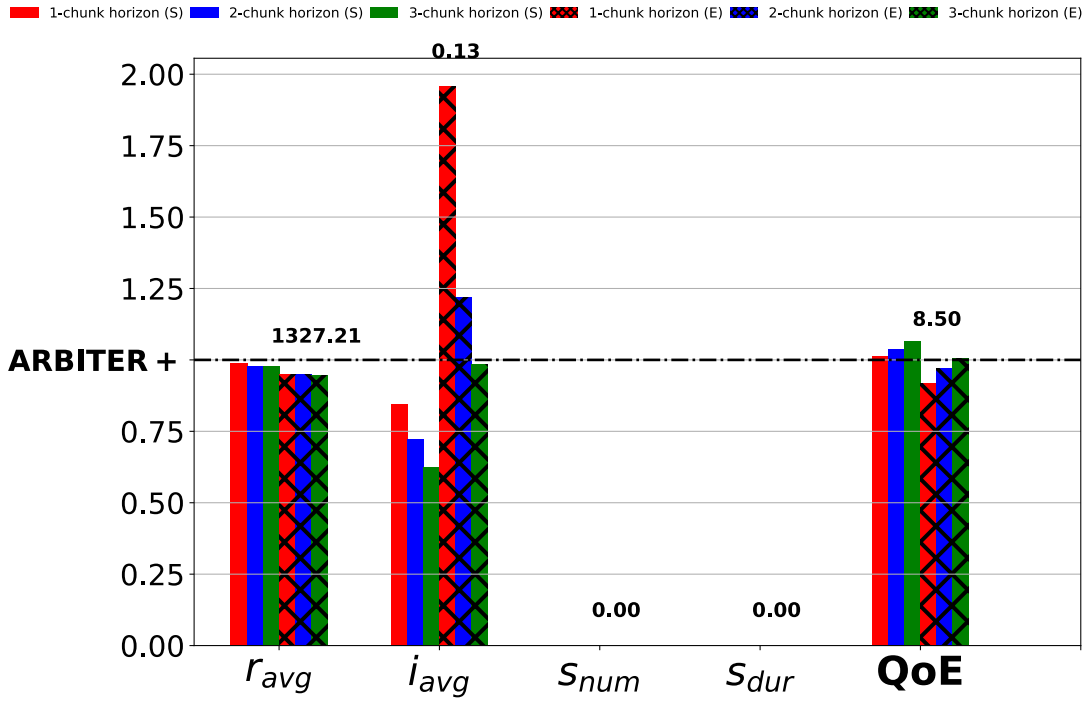


Figure 4.9: ARBITER+: Performance evaluation of QoE metrics for low-variable bandwidth traces (*The metrics are normalised to the no-prediction case with numbers above the bars representing the value of each metric for the no-prediction case*)

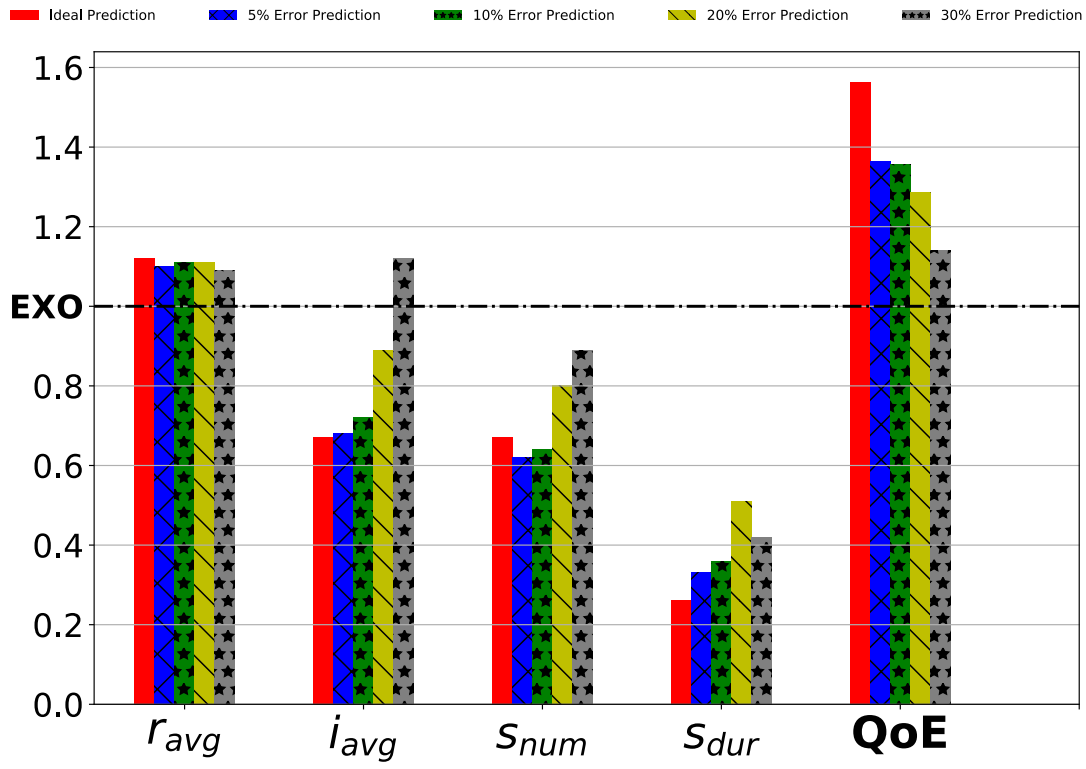


Figure 4.10: Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (EXO)

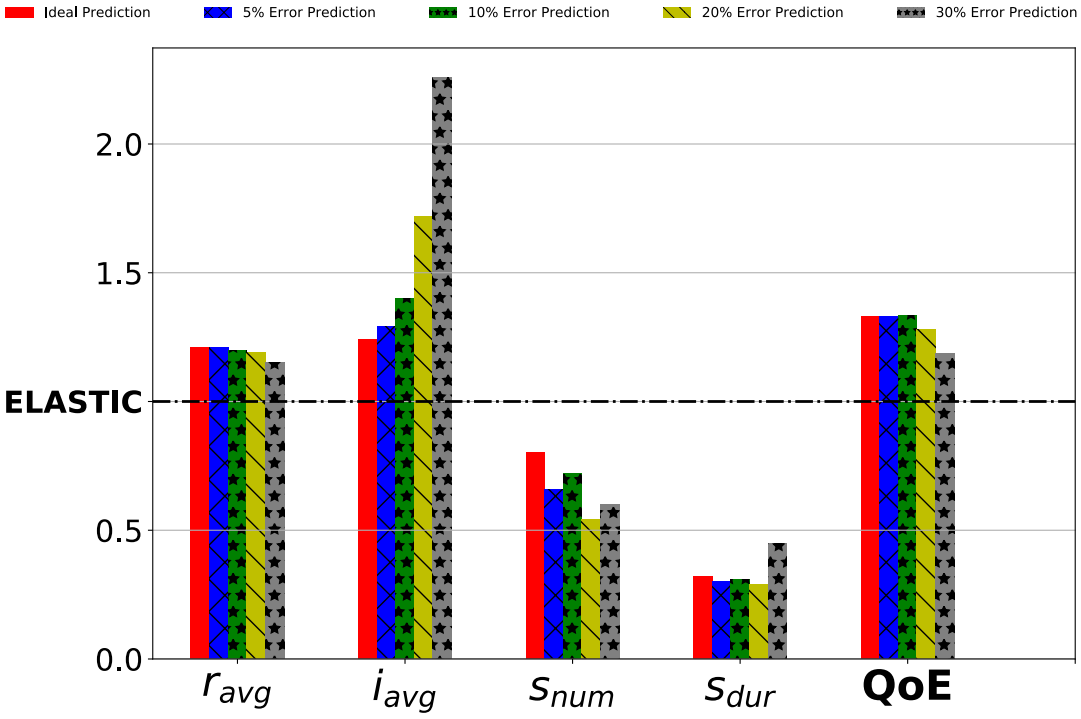


Figure 4.11: Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (ELASTIC)

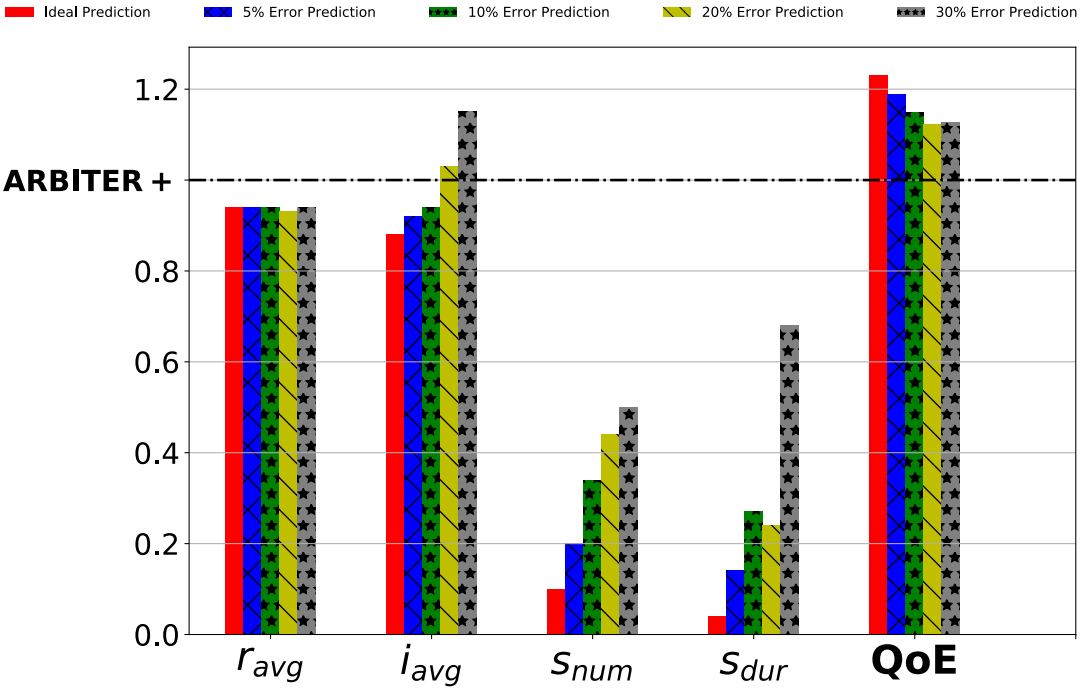


Figure 4.12: Performance evaluation of QoE metrics for high-variable bandwidth traces using error-induced predictions (ARBITER+)

tious approach by insisting on minimising stall events). On the other hand, EXO relies heavily on rate estimation as the additional module that monitors buffer occupancy is only activated for chunk replacement function. As a consequence, error-induced prediction doesn't cause high QoE drop. Still, applying prediction with the significantly high errors (30%) provides overall higher QoE than in *no-prediction* case. The average increase in QoE across all algorithms with the 30% prediction error is 15%. While this result may be surprising, there are three leading causes of this. First, algorithms use predicted value as a guide and apply additional logic limiting the overall impact of prediction. Second, algorithms operate on a discrete set of bitrate rates. On average, the distance between subsequent rates is 30%. As long as the average error is below this threshold, the algorithm will have a significant boost in overall QoE. Third, bandwidth estimators produce significantly higher prediction errors (around 50% and more, depending on bandwidth estimator type).

4.4 Conclusion

This chapter investigated how to integrate throughput prediction with state-of-the-art HAS algorithms and quantify its impact on overall user QoE. The different ways prediction can be delivered to the player's bandwidth estimator, either as a direct estimate or a sample were explored. Furthermore, prediction horizons beyond one chunk duration are explored and examined how different levels of error-induced predictions negatively impact the user experience.

Regardless of the algorithm in use, user QoE improves by a significant 23% in the presence of accurate throughput prediction. Furthermore, the highest QoE is observed in the presence of longer throughput prediction horizons. Most notably, accurate prediction eliminates stall events in an environment with highly fluctuating throughput. While error-induced predictions lower significantly the user QoE in some instances, it still provides a clear 15% gap on average, compared to HAS algorithms with no prediction.

The presented results are very encouraging and motivate the next chapter. The next challenge is moving from "offline" analysis to obtaining accurate predictions in real-time. By leveraging channel information, the next chapter introduces a machine learning framework for throughput prediction.

Chapter 5

Empowering Video Players with Machine Learning based Throughput Prediction in Cellular Networks

The previous chapter motivated the need for accurate throughput predictions, showcasing the significant positive impact on HTTP adaptive streaming (HAS) Quality of Experience (QoE). This chapter presents novel accurate throughput prediction in cellular networks through leveraging Machine Learning (ML) techniques. Also, this chapter builds on results in the previous chapter, performing a similar analysis of HAS algorithms in cellular networks. Furthermore, analysis is extended to a real operational cellular network by implementing the ML model on a real device.

The achievable throughput at any mobile device is coordinated by protocols that operate at diverse time scales, e.g. radio channel scheduling at millisecond-level vs congestion control at hundreds of milliseconds to seconds level. Furthermore, the base station (BS) scheduler allocates the wireless resources based on the bandwidth demand of each device and its channel conditions. While BS schedulers are vendor-specific black boxes, the scheduling algorithms represent a collection of predefined steps and actions and thus exhibits consistent behaviour (i.e. for the same inputs, it will produce the same output). Hence, ML represents an attractive methodology to extract underlying information and correlations in resource scheduling and make accurate throughput prediction for users based on

the abundantly available cellular data.

This chapter makes the following contributions:

- A novel ML-based throughput prediction engine for use by mobile devices in cellular networks. This engine predicts average network throughput for a future time window (horizon) based on radio-specific metrics and network throughput observed over a historical window
- A novel statistical data summarisation method used by the prediction engine that significantly improves the throughput estimation accuracy in comparison to both ML throughput estimation based on raw data samples and application-level bandwidth estimation commonly used in HAS clients. This result illustrates not only the predictive power of radio metrics but also the effectiveness of our summarisation technique.
- Three HAS algorithms are evaluated with and without a ML-based prediction engine. Results are based on 4G cellular traces, real video content and an Android client of a broadly-adopted video player, ExoPlayer. The evaluation considers the impact of essential parameters, such as chunk duration and prediction horizon. In all the tested scenarios, the performance metrics show a clear improvement when using ML, leading to better user QoE.
- To complement the lab-based evaluation, a field evaluation is performed in an operational cellular network. This exercise confirmed results obtained in lab-controlled environment and to identify challenges that merit further research and will be of interest to practitioners seeking to implement ML-based prediction in real systems.

Combining ML with radio channel information allows achieving high prediction accuracy with 90% of errors (the absolute relative difference between actual and predicted throughput) below 18%. All evaluated adaptation algorithms improve their stall (up to 85%) and switching (up to 40%) performance in all tested scenarios. The average quality also improves in most scenarios. Overall, the tested algorithms show improvements of 6%-33% in QoE score.

5.1 Analysis of HAS player architecture design

HAS algorithms commonly consider combinations of network and/or application states for the quality selection decision, as illustrated in Figure 5.1. The *estimator*

module captures the network state, while the *application monitoring* module captures the video player state by monitoring the playback buffer and streamed video quality. The *Estimator* consists of a sample processor sub-module (not shown in the figure) responsible for collecting significant statistics from the *Chunk loader* module in order to estimate the available bandwidth for the chunk. For example, depending on the algorithm design, it can track the rate sample for each chunk or multiple rate samples per chunk. The sample processor feeds the sample array to one of the smoothing functions, from which the final throughput estimate for the next chunk is passed to the *adaptation logic* module. Finally, the *adaptation logic* module combines information from the *estimator* and *application monitoring* modules to decide on the quality of the chunks to be requested. The *Chunk loader* module carries out the decision from the *adaptation logic* module and requests the next chunk. While most of the literature focuses on enhancing

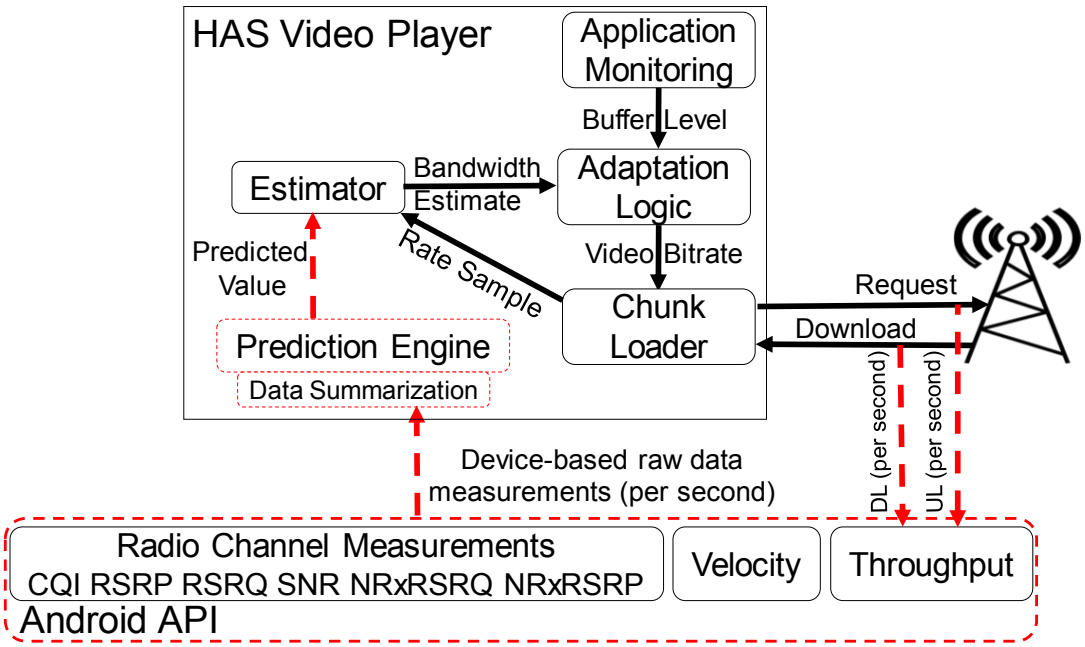


Figure 5.1: Simplified HAS player architecture and addition of the measurement and prediction modules

adaptation logic, little or no work has been done in improving bandwidth estimation. Most of the algorithms employ one of the standard smoothing techniques (e.g. arithmetic, harmonic or exponential moving average to name a few) applied over the n last measured rate samples, where rate samples can be chunk, time or size based. For example, ARBITER+ [ZRS18] combines chunk and time-based sampling with an exponential moving average for bandwidth estimation.

The approach taken in this chapter is to leverage Android Operating System

(OS), which provides Application Programming Interfaces (APIs) for capturing measurements of radio channel metrics, velocity and throughput samples. This capability is used to build a *prediction engine* that captures those metrics at one-second granularity to provide more accurate throughput prediction (see Figure 5.1, dotted boxes). As a result, this module feeds the predicted value to the *estimator* module. The *Estimator* module may process this value through standard smoothing techniques or feed it directly to the *adaptation logic* to improve the decision-making process for the next chunk.

5.1.1 Revisiting bandwidth estimation techniques

Section 4.1 revealed low prediction power of traditional bandwidth estimation techniques in cellular networks. Also, results in Chapter 4 illustrate the positive impact of longer prediction horizons on QoE. To provide a fair comparison, this section extends the analysis from Section 4.1. The analysis is repeated as in Chapter 4 for cases with no throughput guidance. The estimate is compared against a delivery rate of a requested chunk (information from logs is used allowing calculating delivery rates of "future" requested chunks). This case is labelled as "1-chunk". Furthermore, analysis is extended by looking beyond one chunk. For "2-chunk" and "5-chunk" scenarios, the average value of delivery rates is calculated for the next two/five chunks requested. Figure 5.2 depicts accuracy of three classical bandwidth estimation techniques, Exponential Weighted Moving Average (EWMA), average and median. For accuracy, the absolute value of the residual error between throughput estimates and the actual average download rate of chunks is analysed.

The main takeaway from Figure 5.2 is that none of these techniques exhibit high estimation (not prediction) accuracy regardless of how far into the future is looked.

To further illustrate potential gain in improving the *estimator* module, Figure 5.3 shows one randomly selected video session played using Exoplayer and its default HAS algorithm, with and without accurate throughput guidance. With accurate prediction, the player starts at a higher quality and settles to the optimal rate earlier (i.e. highest) than when relying on classical bandwidth estimation (e.g. median). Furthermore, prediction helps to recognise the level of drop in bandwidth more accurately, forcing the client to switch to the lowest quality quickly. As a result, buffer underrun is cut by 90%, improving the overall user experience.

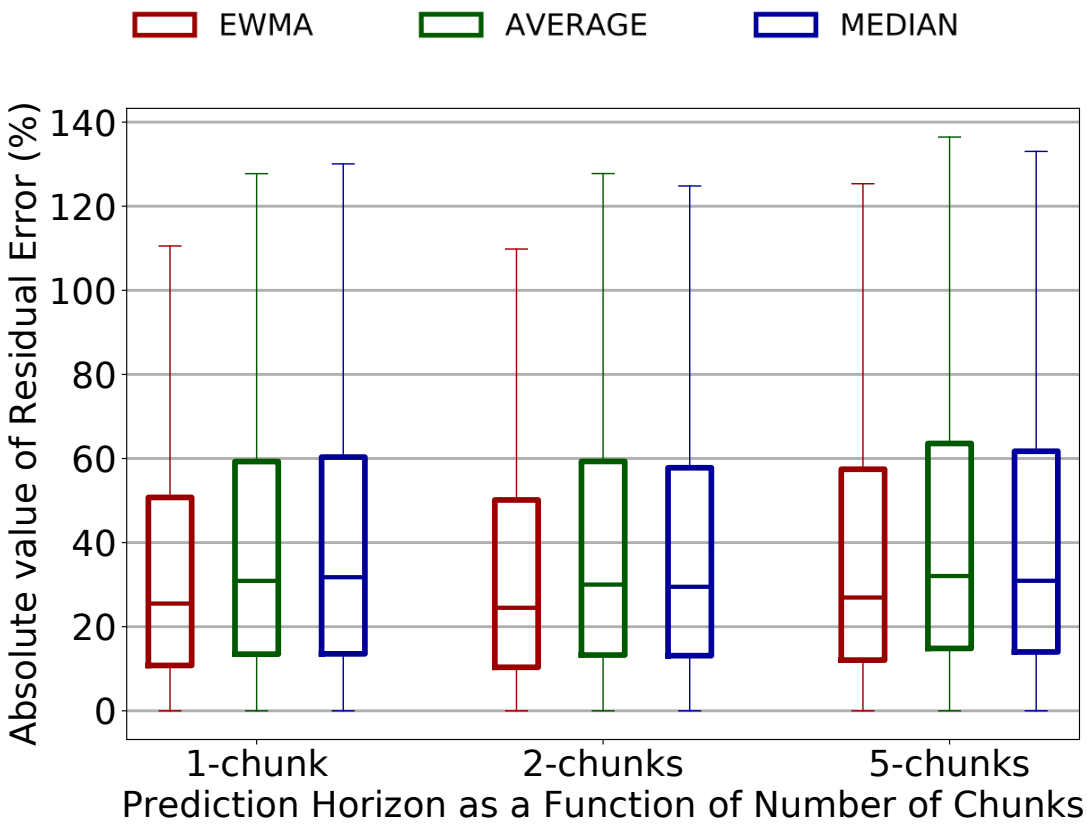


Figure 5.2: Accuracy of different throughput estimation techniques for different prediction horizon values

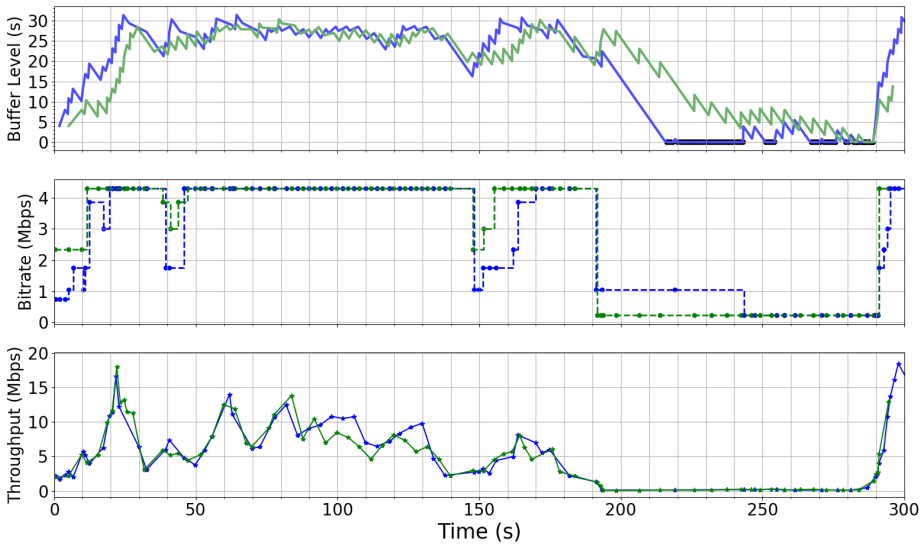


Figure 5.3: Comparing ExoPlayer sessions with throughput prediction (green, 12 sec. horizon) and without (blue)

5.2 Throughput prediction via ML

This section presents the proposed throughput prediction approach and extensively evaluates its performance under different scenarios and configurations. First a possible architecture for the throughput prediction system is outlined. While the majority of work is centred around device-based prediction (explained in the following section), the throughput prediction architecture considers both approaches, including both network and device-based cases.

5.2.1 Throughput Prediction System Overview

There are three main components of the throughput prediction system: the `Collector`, `Predictor` and `Trainer`, as depicted in Figure 5.4. The `Collector` gathers measurement metrics and prepares them for the `Trainer` and `Predictor`. The `Trainer` creates and updates the prediction models using training records that include both data and ground-truth. The `Predictor` translates online collected data records to a throughput estimate.

Figure 5.4 also illustrates different architectural options for each of these components. The `Collector` aggregates a combination of device-level and network-level data through existing interfaces. Device-level data collection is distributed, and hence scalable, but only allows for a local view of the operating context and hence can be less accurate than combined network-level and device-level data. The `Trainer` is typically located in-network (e.g. a datacenter) but can also be located in the device. Training the model in-network can leverage the processing power of datacenters resulting in more robust prediction models. Alternatively, a device-based trainer can pose a bottleneck due to limited computing and power capacities of end-devices. The `Predictor` can run at the device or in-network. A device-based predictor would be usually limited to device-level information but facilitates low prediction delays for end-device applications. Alternatively, a network-based predictor involves both delay and communication overhead but can have access to network-level data. Hence, throughput prediction systems have various architectural choices featuring different advantages and disadvantages. Figure 5.4 illustrates two examples for the architecture of the throughput prediction systems. The first features a device-based collector and predictor with an in-network trainer. The second utilises a full network view with a network-based collector, trainer, and predictor components.

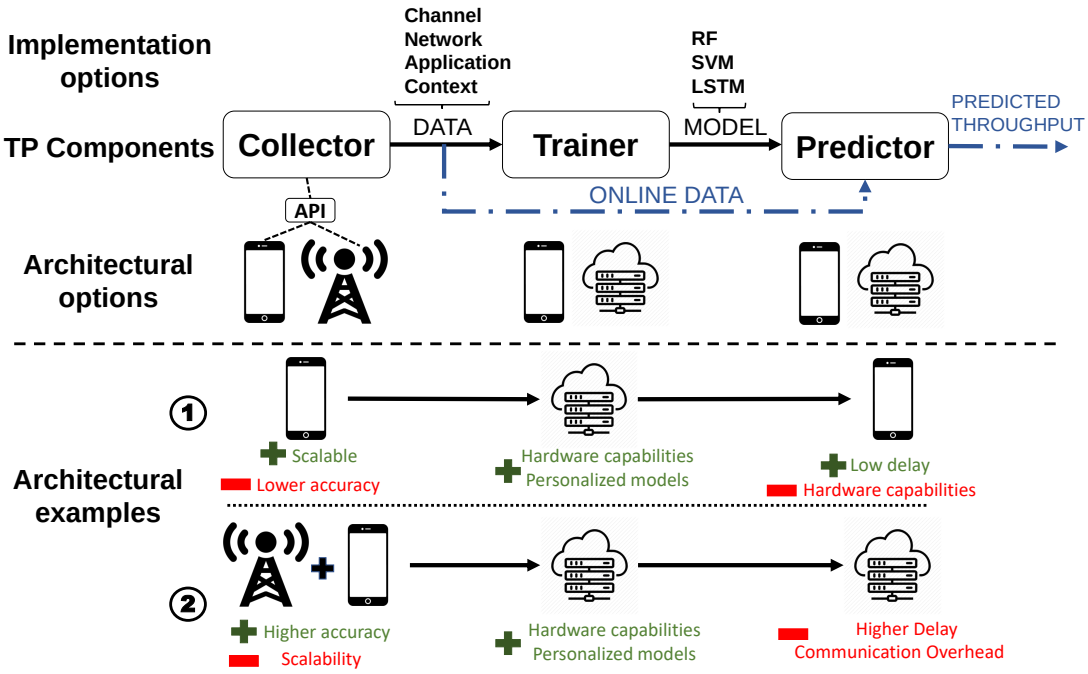


Figure 5.4: Components, implementation, architectural options and examples of AI-driven throughput prediction system

5.2.1.1 Collector

For the `Collector`, the selected metrics and their representation are the key design elements. These metrics can be classified as channel-specific data (e.g. CQI - Channel Quality Indicator), network-specific data (e.g. cell load), application-specific (e.g. application throughput), and context-specific data (e.g. mobility mode).

Device-level data represents an individual device environment and can be collected at the mobile device using its Operating System (OS) Application Programming Interface (API), e.g. Android telephony. However, this approach relies on OS implementation and typically have a medium time resolution, e.g. one-second. While device metrics can be collected at millisecond granularity, this requires specialised software and hardware equipment. In Section 5.5, we further illustrate the impact of higher resolution metrics, collected using unconventional methods, on the prediction accuracy.

Network-level data represents a comprehensive view of metrics for multiple users served by one BS or more BSs. In the current generation (i.e. 4G) of cellular networks, there is no unifying approach for measuring and reporting various network-level metrics. Different vendors use different approaches to collect and

report various metrics, requiring immense effort to consolidate data from various sources.

5.2.1.2 Predictor & Trainer

The `Predictor` and `Trainer` are tightly coupled. For these two components, the selection of the machine learning algorithm is the most crucial design decision.

5.2.2 Proposed prediction technique

In the proposed design, the focus is on the prediction of *average* throughput rather than *instantaneous* throughput. The needs of HAS applications motivate this decision. To illustrate, when a streaming client downloads video chunks, throughput fluctuations over the future x seconds is of little concern for the player. What matters is the average throughput that the video player will observe in the next x seconds, as this will drive the behaviour of the video adaptation algorithm. x is the *prediction horizon* and it is an application specific design parameter. The data driving this prediction includes both radio metrics and throughput samples collected at the device at arbitrary granularity. Granularity of 1 second is used; this is driven by the sampling period used in capturing radio data as explained in Chapter 3.1. However, the proposed prediction technique can also be used in conjunction with other sampling time scales. A typical prediction engine input would consist of one or more historic radio metrics and throughput samples. In the following sections, several *scenarios* for the combination of prediction horizon and history length are investigated. The notation used is *PyFx*, where Py denotes the past y seconds of historical data, and Fx denotes a prediction horizon of x seconds. In Section 5.2.5, the impact of varying Py and Fx on the prediction accuracy used is shown.

The combination of channel conditions and the current state of a cell largely determines the number of allocated radio resources blocks; this translates to achievable throughput at the device. To capture these dimensions, the following radio channel metrics in conjunction with physical mobility speed (in kph) and historical application throughput are used (more details about each metric can be found in Chapter 3.1):

- *Reference Signal Received Power (RSRP)*
- *Reference Signal Received Quality (RSRQ)*

- *Signal to Noise Ratio (SNR)*
- *Channel Quality Indicator (CQI)*
- *NRxRSRQ, NRxRSRP*
- *Downlink Throughput/ Uplink Throughput*

Random Forest (RF) is used as the ML algorithm. RF [Bre01] represents an ensemble/boosting learning method for regression and classification tasks. The main reasons for selecting the RF are the following:

1. RF works by growing a collection of decision trees (weak learners) and then making a prediction by taking the mean of individual trees. This approach reduces overfitting because each tree is constructed on a randomly selected subset of features. They are further de-correlated (which minimises the overfitting) by considering a random subset of features for each split of the decision tree.
2. RF can be used for feature ranking, enabling analysis of the importance of each metrics used for training. Finally, it requires minimal tuning (the number of trees is the most critical hyperparameter). Also previous studies show that RF outperforms other ML techniques for this problem space [TKV18, RZS+17].

5.2.2.1 Quantile summarization techniques

When applying machine learning techniques, the first step includes feeding data without any processing (except normalization methods). For throughput prediction, the full *history* of each metric (*raw* in the list below) is used. However, classical ML algorithms usually require high-level features extracted from raw data (i.e. feature engineering) to achieve high accuracy. For example, history may be summarised, e.g. by the average value. However, the average and entire history can be affected by outliers and can react slowly to changes if maintaining a long history. Having outliers in real data is an unavoidable hurdle. For throughput prediction, capturing patterns based on historical data is the main interest. Instead of feeding every sample collected at the arbitrary time interval to the ML algorithm, only key values that best summarise the data are fed. Collected historical data represents a distribution for each metric. To infer unknown distribution from empirical data, percentiles are used (if historical data follows a normal distribution, using mean and standard deviation is enough to

explain the whole distribution). Different combinations of percentiles (including the mean) are tested empirically. Finally, the inter-quartile range, mean and 90^{th} percentile give the highest prediction accuracy and are selected in further analysis. Motivated by this, the following *Quantile* summarization techniques is proposed; for a $(m^n_{i-1}, m^n_{i-2}, m^n_{i-3}, ..)$ array of values we calculate the following metrics: $25^{th}, 50^{th}, 75^{th}, 90^{th}$ and *mean* of the input array, where m^n_i represent n_{th} metric at time i.

This summarization technique is compared to the strawman prediction based on *raw* data;i.e. using $(m^n_{i-1}, m^n_{i-2}, m^n_{i-3}, ..)$ for every n as input to the prediction model.

5.2.3 Evaluation of the prediction techniques

5.2.3.1 4G Dataset

Evaluation is based on the previously collected 4G dataset with Radio Access Network (RAN) metrics, see Chapter 3. Traces reflect the use of the Android API by providing previously mentioned device-level metrics. The collected trace profiles are a mixture of five different mobility patterns: *static*, *pedestrian*, *bus*, *train*, *car* and *highway*. In [YJS+18], the authors show 500 records are enough for successfully training an ML model (authors use similar RF algorithm). Following their result, we filter out all traces with less than 500 records.

5.2.3.2 Metrics

The Quantile and raw ML-based throughput prediction techniques described above are compared using two key metrics: the *absolute value of residual error* (ARE) and *coefficient of determination* (R^2).

ARE is the ratio of absolute residual error and actual throughput, where the residual error is the difference between actual and predicted throughput. The following equation defines ARE:

$$ARE = \frac{|max(10, R_i) - max(10, \hat{R}_i)|}{max(10, R_i)} \times 100 \ , \tag{5.1}$$

where R_i and $\hat{R}_i$ is the actual measured throughput and predicted (estimated) throughput (in Kbps), respectively. To avoid cases when actual bandwidth drops

to zero causing zero division in equation 5.1 we bound all values less than 10Kbps to 10Kbps.

R^2 score (0-1) is a measure of the goodness of a model compared to a naive model. e.g. a R^2 score of 0.8 (respectively 0.9) implies that the naive model has five (respectively ten) times higher error than the model in question. The following equation defines R^2:

$$R^2 = 1 - \frac{\sum_{i=0}^{N}(R_i - \hat{R}_i)^2}{\sum_{i=0}^{N}(R_i - \bar{R})^2} \ , \tag{5.2}$$

where N is the number of samples in the test dataset, and $\bar{R}$ the average throughput for the test dataset.

When evaluating performance of RF, it is crucial to analyse generalisation error (also called out-of-sample error), which represents the model's ability to represent *unseen* data. Generalisation error is a composition of bias, variance, and noise. Bias represents error introduced by approximation of a more complex problem with a simpler model. In other words, if the model assumes a certain relationship for data which does not reflect the actual data generation mechanism, bias exists in estimate. To capture this, training error (i.e. ARE metric for our specific case) is used. However, low training error does not necessarily translate to low bias. Low training error could be a consequence of limited data that do not entirely capture the nature of actual problem. In this case, you could have low training error but significant bias. Variance represents the variability of error as we vary the dataset (e.g. re-sampling the training set). Generalisation error is quantified using Cross-Validation (CV) data to test a model. If the resulting error is low, then the model is said to have a small generalisation error, implying that it can successfully predict new values on unknown data. A high error indicates that the model *overfits* the training data, and is thus only capable of predicting values based on data similar to the training data. The desirable outcome is to have both low training and CV errors.

Unless otherwise noted, the RF algorithm is tuned and estimates its quality using 10-fold cross-validation[1]. Cross-validation is computationally more expensive than alternative techniques like "holdout", but it guarantees higher accuracy [HTF09].

In the following evaluation, a large set of parameters are explored (e.g. horizon, history length, and summarization techniques). Furthermore, a majority of

[1] *sckit-learn* tool is used as the main ML framework (https://scikit-learn.org)

experiments are done in a mobile case (**highway** scenario), as it is the most challenging environment (see section 5.2.6). Hence, a *funnel-based* approach is used where progressively some parameters are fixed after investigating their impact on throughput accuracy as presented in the following subsections.

5.2.4 Impact of the Quantile summarization technique

Figure 5.5, 5.6 and 5.7 show the ARE for two approaches: feeding raw input, and applying the quantile summarization technique to the input data. These two approaches are compared using various combinations of PxFy; i.e. history and horizon duration.

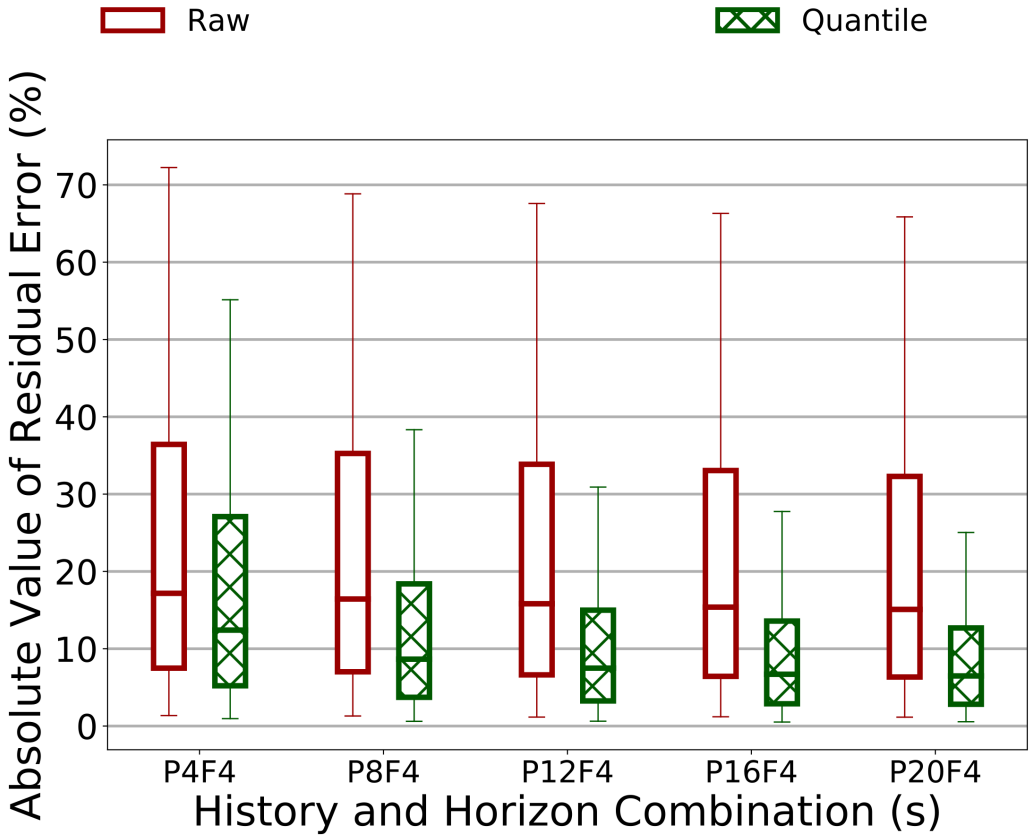


Figure 5.5: ARE values as a function of history (horizon = 4 seconds)

Regardless of the prediction horizon, the proposed quantile summarization technique achieves lower ARE (higher prediction accuracy) than the raw technique. This difference gets bigger with a longer metric history. For example, with a prediction history of 20 seconds (P20Fx) the quantile technique lowers the ARE compared to raw by 20% (75^{th} percentile) and 40% (90^{th} percentile). For R^2

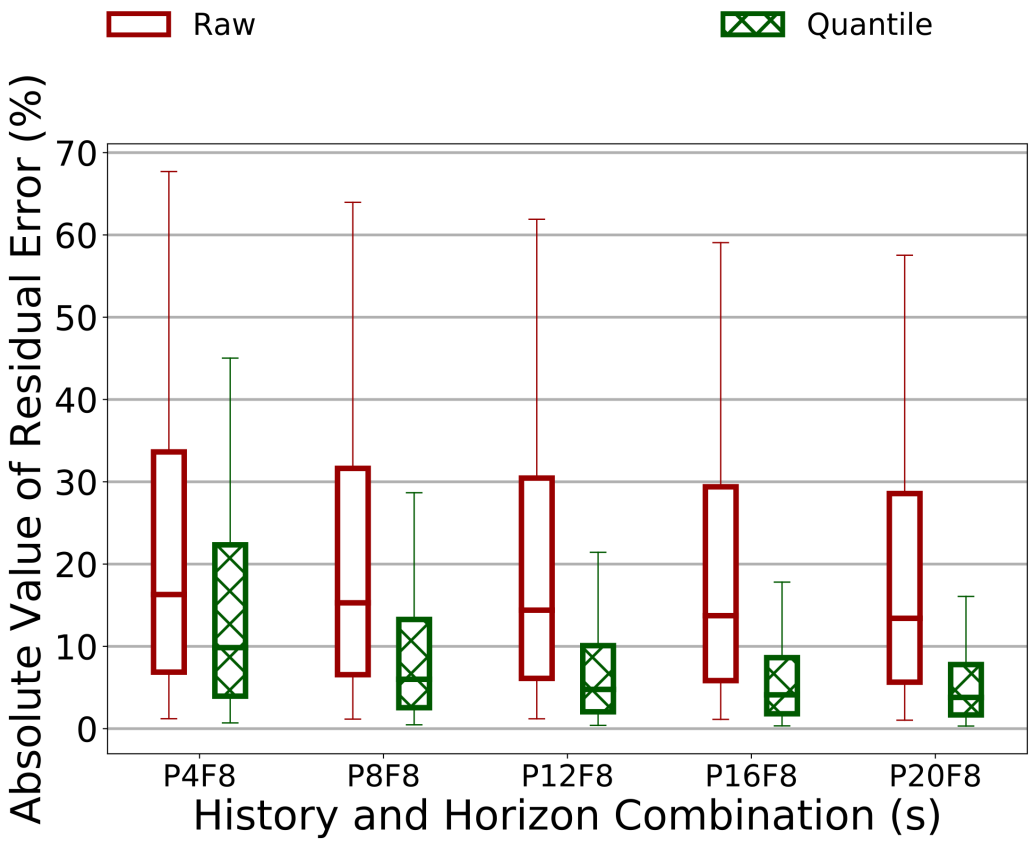


Figure 5.6: ARE values as a function of history (horizon = 8 seconds)

score, we observe a similar trend as for ARE, e.g. a value of 0.99 for large history when using the quantile strategy which is a 0.05 boost compared to the raw approach.

To understand the causes of different prediction accuracy across different history lengths, the standard approach of analysing the learning curve is used. The learning curve represents a ratio/difference between training and cross-validation error metric. The choice of error metric is arbitrary, as more emphasis is on the difference obtained from training and CV data. For the following analysis, Coefficient of Determination (CoD) is chosen for the error metric. Next, both training and CV error are investigated (see Section 5.2.3.2) for RF. Table 5.1 shows both training and CV error for the RF algorithm above as a function of the history length, i.e. amount of training data considered. RF does not suffer from high bias for any history and horizon combinations. However, for history lengths shorter than eight seconds, the RF model relatively overfits the training data. This effect is countered as the history length increases.

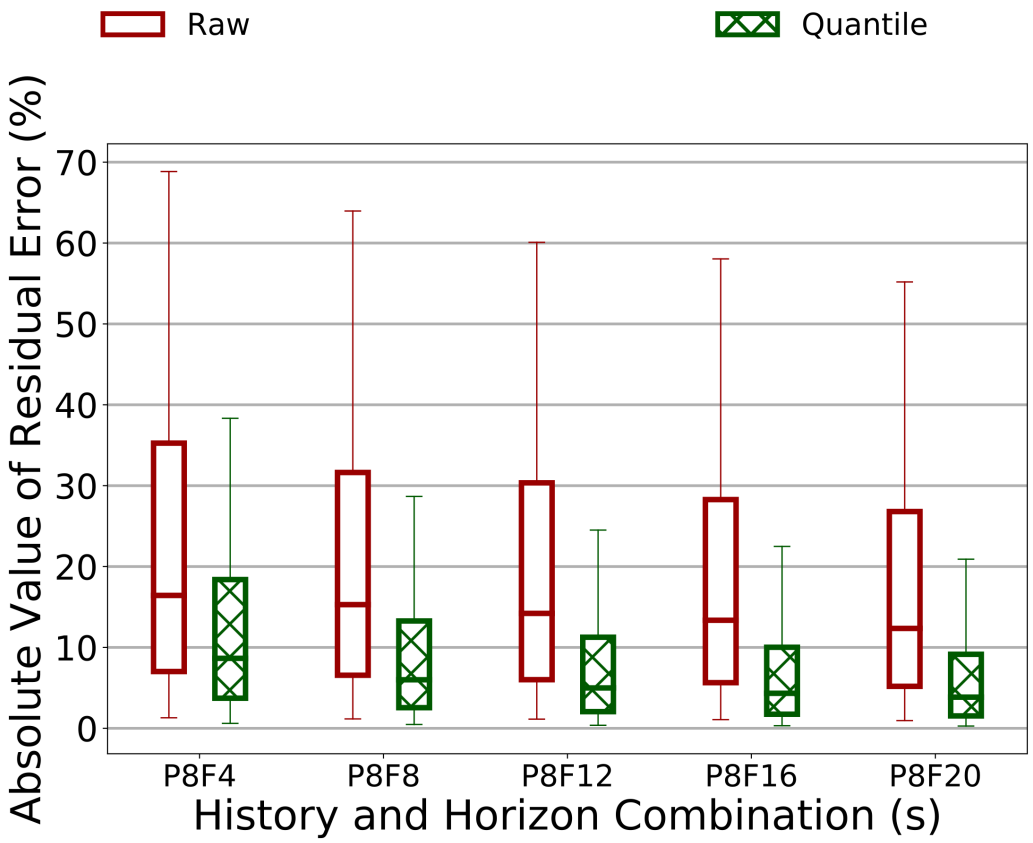


Figure 5.7: ARE values as a function of horizon (history = 8 seconds)

Table 5.1: Learning Curves for RF algorithm as a function of history interval and 12-second horizon

Px =	**4s**	**8s**	**12s**	**16s**	**20s**
Train sc.	1.0	1.0	1.0	1.0	1.0
CV sc.	0.93	0.97	0.98	0.99	0.99

5.2.5 Impact of prediction horizon and history length

Figure 5.7 also shows that throughput prediction accuracy improves when increasing the prediction horizon. At first, this result appears counter-intuitive as one would expect that predicting throughput for a near future should be easier than for the more distant future. This observation is true for classical estimation techniques, such as EWMA, AVERAGE, and MEDIAN (Figure 5.2) which rely solely on past throughput values and coarse granularity of history samples to estimate future capacity.

However, when predicting the *average* throughput over the next x seconds by using both radio metrics and throughput, overall accuracy improves with the longer horizons. Reason for improved accuracy is that larger horizon results in

variance decrease among throughput values. Figure 5.8 illustrates the effects of averaging over different horizons resulting in a flatter throughput curve.

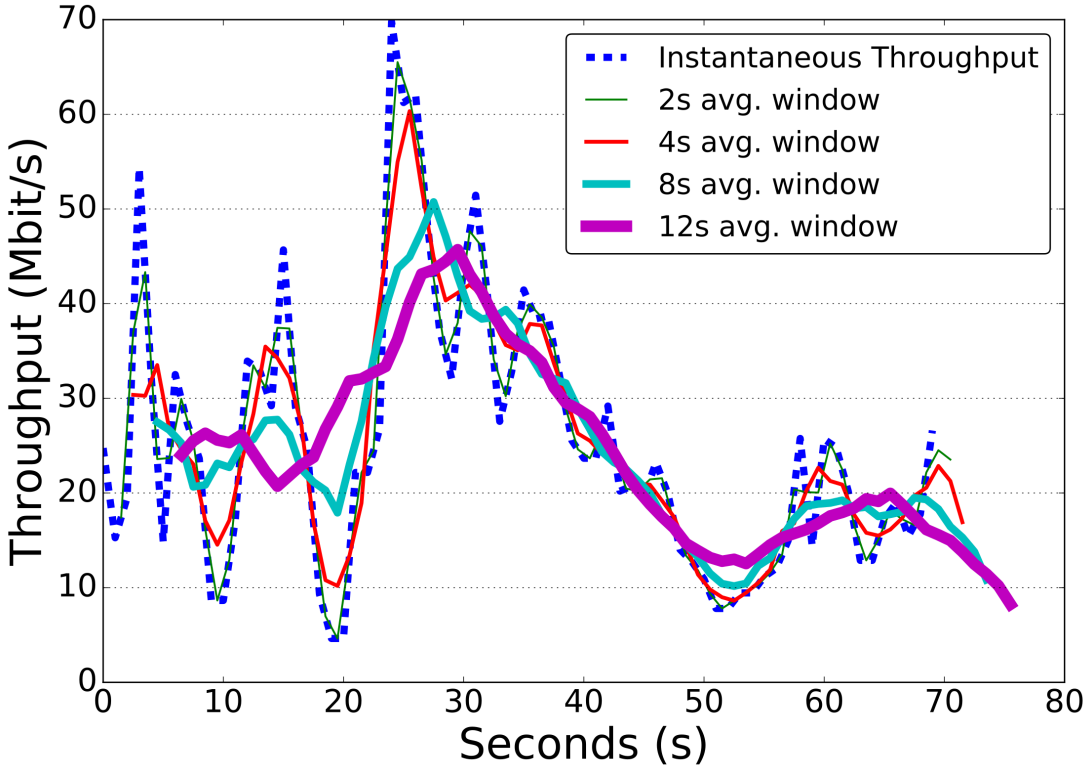


Figure 5.8: Prediction horizon analysis

To underpin above observation, mathematical analysis is performed. Traditional and well-established notation is used in describing traces as sample realisations from an infinite population of time series generated by the stochastic process.

Let x_i be measured samples of instantaneous throughput. New samples of process F^k is defined as:

$$f_i^k = \frac{1}{k}\sum_{j=1}^{k} x_{i+j} \quad \text{for } i \in [1, N-k] \tag{5.3}$$

where N is number of samples in trace, and k is averaging window size. The average value of the transformed trace is:

$$\mathbb{E}(F^k) = \frac{1}{N-k}\sum_{i=1}^{N-k} f_i^k \tag{5.4}$$

Average value would depend on properties of measured time series. Using Augmented Dickey-Fuller [Wil16] test, the majority of our traces have properties of stationary stochastic processes. This property results in values oscillating about a constant mean across all time points.

Using this feature, (5.3) results in:

$$f_i^k = \frac{1}{k}\sum_{j=1}^{k} x_{i+j} \approx \mu \tag{5.5}$$

where $\mu = \frac{1}{N} \times \sum_{i=1}^{N} x_i$. Using (5.5) in (5.4):

$$\mathbb{E}(F^k) \approx \mu \tag{5.6}$$

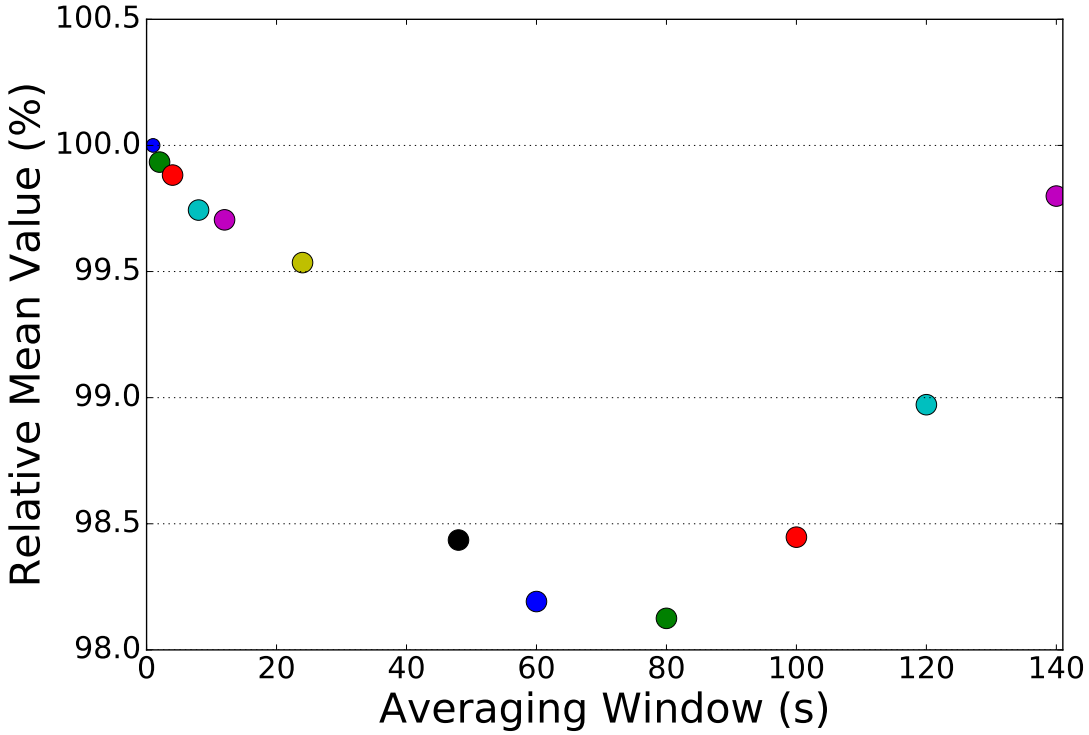


Figure 5.9: Relative average throughput values across different intervals (real trace)

Figure 5.9 depicts relative average value for different averaging windows as a ratio between average instantaneous throughput and average throughput for a given interval. Average value fluctuates around reference value depending on window size. Maximum difference is less than 2%, and for cases of interest (2s, 4s, 8s, 12s) average values differ by 0.5%. This concludes that (5.6) holds, and mean value does not significantly change with window size.

Next, variance of process F^k is defined as:

$$Var(F^k) = \frac{1}{N-k}\sum_{i=1}^{N-k}(f_i^k - \mu)^2 \tag{5.7}$$

For each sample f_i^k:

$$(f_i^k - \mu)^2 = \frac{1}{k^2}[\sum_{j=0}^{k-1}(x_{i+j+1} - \mu)^2 \tag{5.8}$$

$$+2\sum_{j=0}^{k-2}\sum_{z=j+1}^{k-1}(x_{i+j+1} - \mu)(x_{i+z+1} - \mu)] \tag{5.9}$$

Term $\sum_{j=0}^{k-2}\sum_{z=j+1}^{k-1}(x_{i+j+1}-\mu)(x_{i+z+1}-\mu)$ represents an auto-covariance. For uncorrelated data, this term equals to zero. Assuming that data is uncorrelated (5.8) becomes:

$$(f_i^k - \mu)^2 = \frac{1}{k^2}\sum_{j=0}^{k-1}(x_{i+j+1} - \mu)^2 \tag{5.10}$$

Combining (5.10) and (5.7) leads to:

$$Var(F^k) = \frac{1}{N-k}\frac{1}{k^2}\sum_{i=1}^{N-k}\sum_{j=0}^{k-1}(x_{i+j+1} - \mu)^2 \tag{5.11}$$

Using $\sum_{i=1}^{N-k}\sum_{j=0}^{k-1}(x_{i+j+1} - \mu)^2 \leq k\sum_{i=1}^{N}(x_i - \mu)^2$ in (5.11):

$$Var(F^k) \leq \frac{N}{N-k}\frac{1}{k}Var(X) \tag{5.12}$$

For large N variance of transformed process decreases as we increase averaging window size.

However, data is correlated, as depicted in Figure 5.10.

Next, (5.12) becomes:

$$Var(F^k) \leq \frac{N}{N-k}\frac{1}{k}Var(X) + 2\frac{1}{N-k}\frac{1}{k^2}\sum_{i=1}^{N-k}\sum_{j=0}^{k-2}\sum_{z=j+1}^{k-1}(x_{i+j+1} - \mu)(x_{i+z+1} - \mu) \tag{5.13}$$

The total variance of process F^k now also depends on the value of auto-covariance. However, covariance will be reduced by the k^2 factor. Expectation is that in majority of cases the resulting variance will be lower than $Var(X)$.

Figure 5.11 illustrates relative variance for different averaging window sizes. Relative variance is defined as the ratio between variance for "original" time-series and variance of transformed time-series with averaging interval k. Variance drops

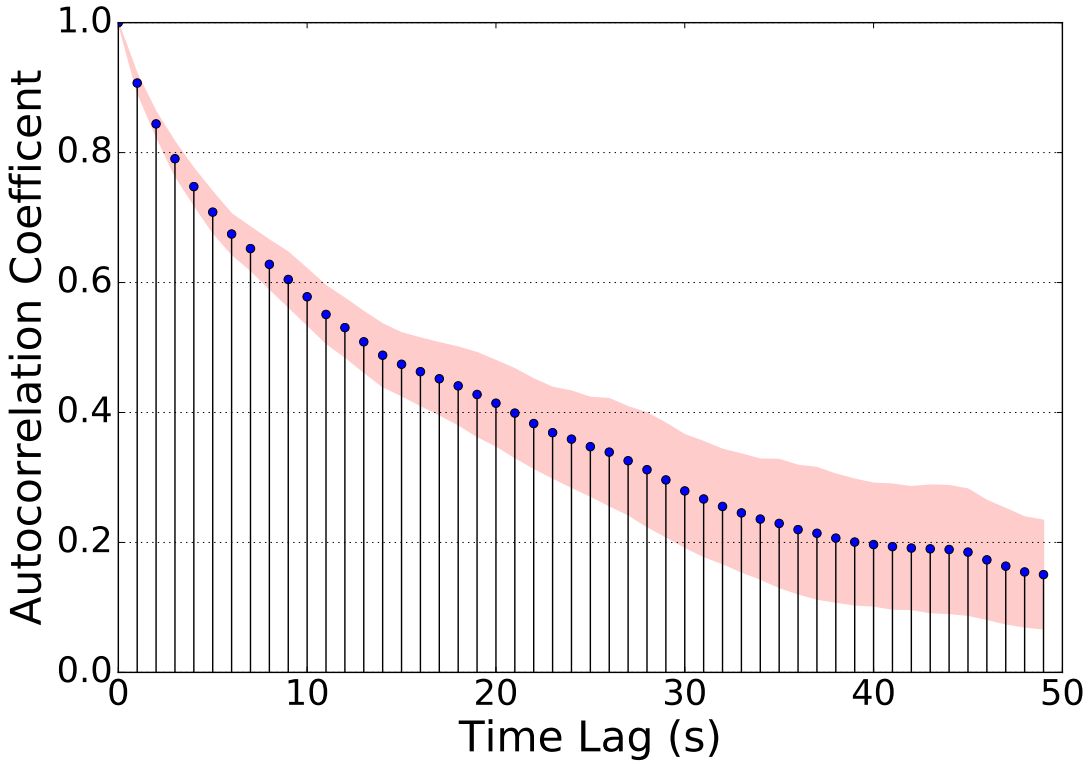


Figure 5.10: The average ACF of throughput vs. time lag across all traces

as we increase k as expected. Variance drops by 3.5%, 7%, 12%, and 16.5% for 2s, 4s, 8s, and 12s respectively. In conclusion, variance significantly decreases as averaging window size is increased.

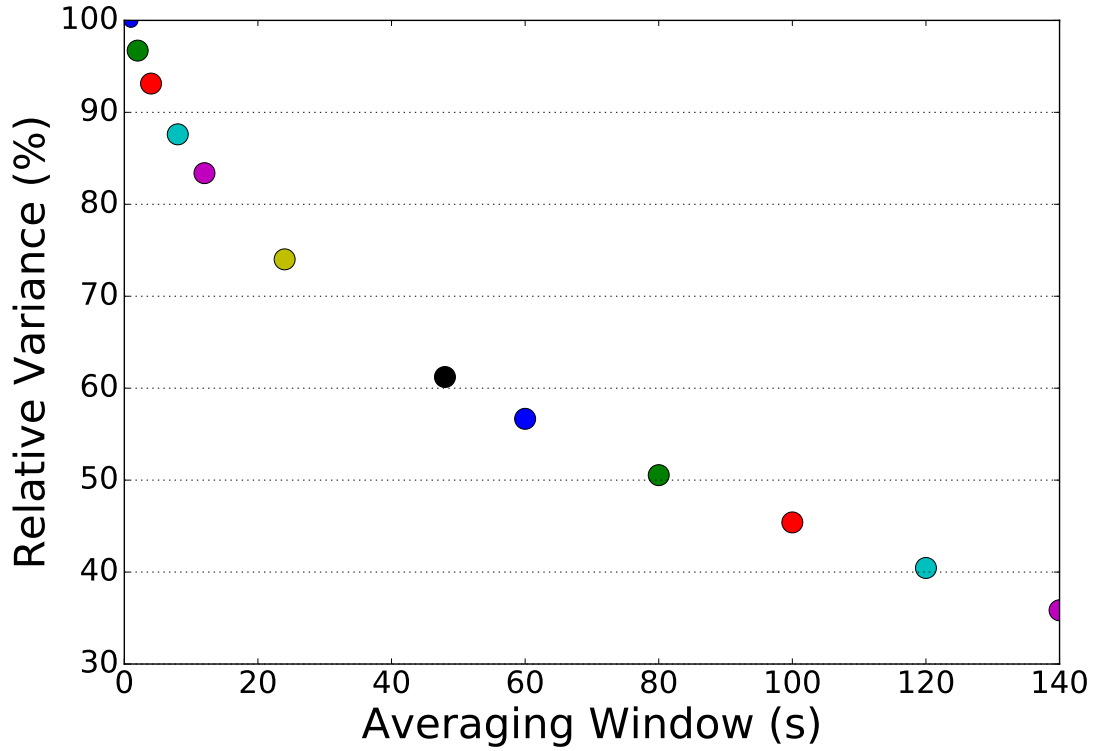


Figure 5.11: Relative variance of throughput values across different intervals (real trace)

Similar to the increasing horizon, analysis also suggests a decrease of ARE as the history length increases. To confirm this, Figure 5.12 shows the ARE as a function of increasing history length. The figure shows that increasing history length is beneficial in term of ARE reduction up to a *saturation point* of 20 seconds, beyond which the ARE reduction is marginal. Furthermore, similar observations hold for 4 and 8-second horizons. The same trend can also be seen for the R^2. In [YJS+18] authors show that increasing history decreases prediction accuracy countering results of our own. Authors argue that having large history results in predictor having less power in reacting to sudden changes in the wireless channel. This intuition comes from the inability of sudden changes to be reflected when averaging over a large history period. However, having multiple measures of variability helps in countering this effect. e.g. 25^{th}, percentile can capture small changes in link variability. Based on this result, in the following, history length up to 20 seconds is considered.

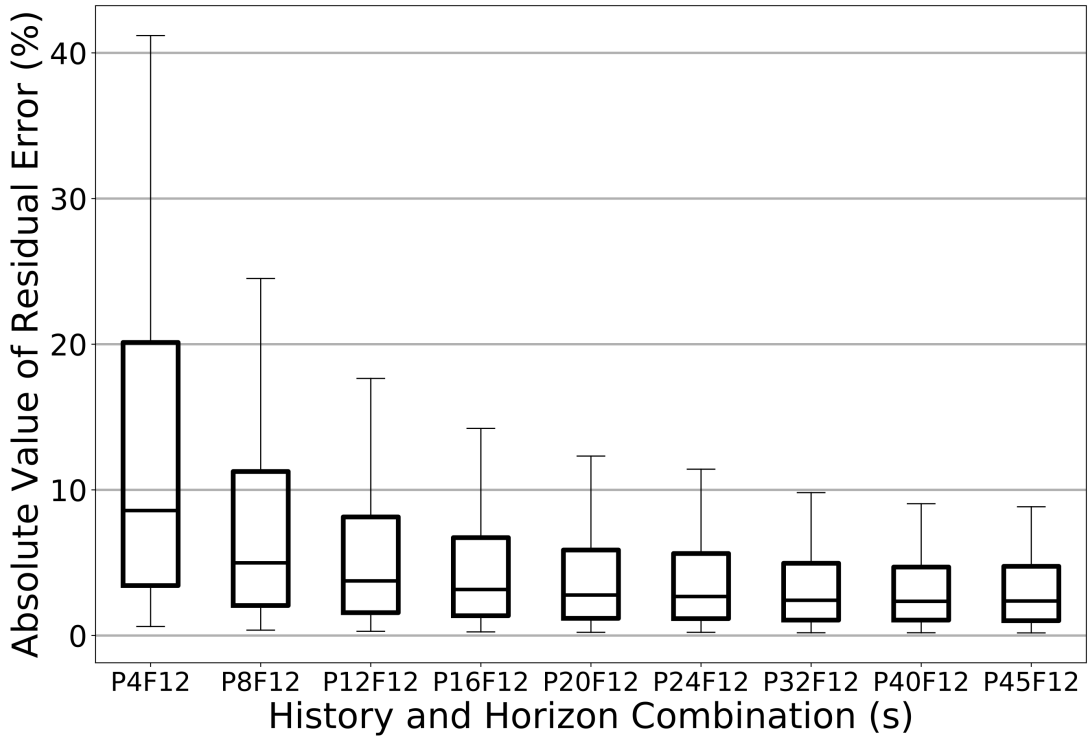


Figure 5.12: ARE for 12s horizon and varying history length

5.2.6 Impact of different mobility patterns

The performance of proposed ML technique is analysed across different mobility patterns. Table 5.2 shows average, standard deviation and Coefficient of Variation (CoV) for all five mobility patterns. Intuitively, a static case has the lowest

Table 5.2: Throughput stats for various mobility patterns

Mobility Pattern	Mean (Mbps)	Std (Mbps)	CoV
Static	12.37	5.04	0.4
Pedestrian	10.22	7.59	0.74
Bus	13.97	12.35	0.88
Car	16.29	12.53	0.77
Highway	14.56	12.65	0.87

standard deviation. We expect that static case should yield higher prediction accuracy due to lower variation in throughput values. The larger standard deviation implies a throughput time series that is less "stable" around the mean. The higher variability of the mobile scenarios (we classify bus, pedestrian, car, and highway as mobile cases) is due to *environmental* changes, e.g. channel and cell load. Intuitively, predicting a throughput with lower variation is an easier challenge; following analysis quantifies this observation further.

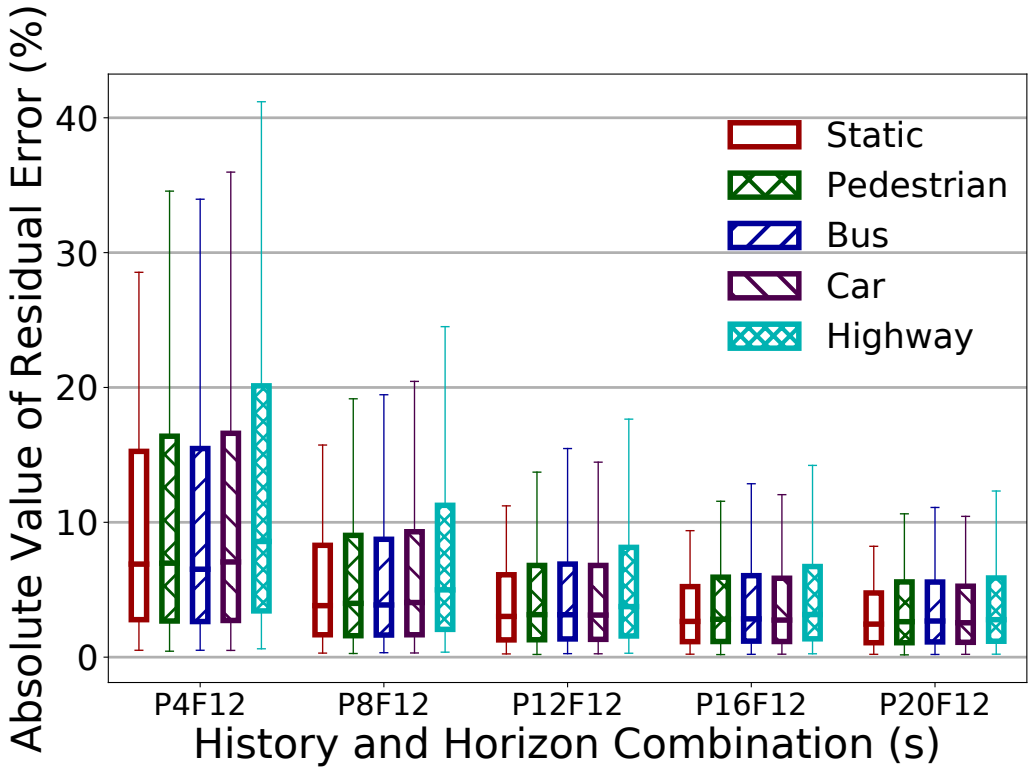


Figure 5.13: Comparison of ARE for various mobility use-cases (PxF12)

Figure 5.13 shows ARE values for static and mobile use-cases. The prediction horizon is fixed to 12 seconds but vary the history duration. Overall, the figure shows much better accuracy (lower ARE) in the static scenario. The influence of history length on accuracy for static and mobile cases is compared. With a history length of 4 seconds, 90% of time prediction error is less than 30%, for static case, while for the highway case this increases to 43%. However, extending history to 8 seconds (4 seconds is used as a threshold to exploit the full benefits of the *quantile* approach), benefits both mobile and static cases, as the 90^{th} percentile of ARE drops by 20% on average for different mobile cases, while in static scenarios this drop is 16%. Increasing history follows the same trend, e.g. 90^{th} of ARE decreases by 40% and 46%, for the mobile and static case, respectively. Nevertheless, the pattern changes for history length beyond 12 seconds, as relative error difference becomes more prominent (e.g. 20-second history lowers 90^{th} percentile by 74% and 71% for static and mobile, respectively). Among mobile cases, overall *highway* scenario shows the highest prediction error. However, the difference between different mobile patterns is negligible.

Similar trends are observed for other values of prediction horizon, e.g. for P20F8 the 90^{th} of ARE for static and mobile cases is 12% and 16%, respectively. Sim-

ilarly, for P20F4 the 90^{th} of ARE for static and mobile cases is 19% and 26%, respectively.

The key conclusion is that utilisation of the quantile technique allows high prediction throughput accuracy for longer prediction horizons, regardless of the mobility environment.

5.2.7 Metric contribution to prediction accuracy

The importance of metrics in throughput prediction is investigated next. Instead of reporting individual metrics (the reader is referred to [YJS+18] for analysis of each feature individually), metrics are divided into three groups reporting the importance of each group. Each group represents a distinctive set of features. For example, CQI, RSRP, and SNR are all features related to channel characteristics. In the same vein, uplink and downlink throughput is grouped separately.

Metrics are divided into the following groups: *throughput* (which includes the history of both downloads and uploads throughput values), *radio* (which consists of the history of RSRP, RSRQ, SNR, etc.) and device *velocity*.

Table 5.3a shows how feature importance changes as we vary the history length. For the P4F4 case, historical throughput contributes to 71% of future throughput prediction, and radio metrics and velocity contribute 25% and 4%, respectively.

With an even longer history, the quantile approach can finally be applied, getting a greater contribution from radio metrics. As the history increases from 2 to 20 seconds, radio metrics importance increases to 41%, while throughput importance drops to 53%. Table 5.3b shows that for fixed a history length, throughput

Table 5.3: Feature Importance for PxF4 and P4Fx cases

(a) PxF4

	P4F4	**P8F4**	**P20F4**
Radio	25%	31%	41%
Throughput	71%	65%	53%
Velocity	4%	4%	6%

(b) P4Fx

	P4F4	**P4F8**	**P4F12**
Radio	25%	32%	36%
Throughput	71%	62%	57%
Velocity	4%	6%	7%

importance goes down with longer horizons. For example, for P4, the importance of throughput goes down from 71% for F4 to 57% for F12. The drop is significant, and similar trends are observed for other values of history length as well.

For example, in the P20Fx scenario, throughput importance drops from 53% to 48% to 44% for 4-second, 8-second, and 12-second horizon, respectively.

Finally, when predicting for longer horizons, the ML model learns more on non-throughput metrics for making a more accurate prediction. This result explains why relying only on throughput is not a good indicator of distant throughput in a highly mobile scenario.

5.2.8 Mobility impact on prediction accuracy

Many factors skew prediction accuracy, including the choice of ML algorithm, radio metrics and random outliers coming from real measurements. In evaluation, boxplot notation is used to counter occurrence of outliers. However, in the mobility scenario, there is an additional limiting factor that cannot be addressed with the current model. Figure 5.14 shows ARE and RSRP values for a mobile device moving between two cells. The prediction case with the 12-second horizon is analysed. The black dotted line represents a handover event. RSRP is selected as it is a good indicator of edge conditions. The RSRP represents an average over future 12-seconds and thus matches the prediction horizon. It is clear that the error is relatively low except around the cell-edge region. As the device approaches the edge region, RSRP sharply drops while error increases significantly. There are two main reasons for this result. When predicting future values, the prediction model relies on past as the input. However at the time prediction is made, current channel metrics have relatively high values indicating good channel conditions. Nevertheless, as shown in the figure, RSRP drops suddenly due to device mobility.

Next, for all traces, records are split based on calculated future RSRP into two categories: records with RSRP larger than -100 are grouped and vice versa. This value is chosen to extract the edge region around cells. Finally, Cumulative Distribution Function (CDF) of ARE is plotted for two cases, as depicted in Figure 5.15. For the records with RSRP smaller than -100, CDF is left of the case with larger RSRP, indicating overall higher errors.

One possible solution to counter higher errors in the cell-edge region is to use a shorter history. However, this approach would result in overall higher ARE. On the other hand, enhancing the dataset by adding new features related to edge conditions is another approach. For example, geographical distance between the current serving cell and all neighbouring cells could be added. Then the prediction

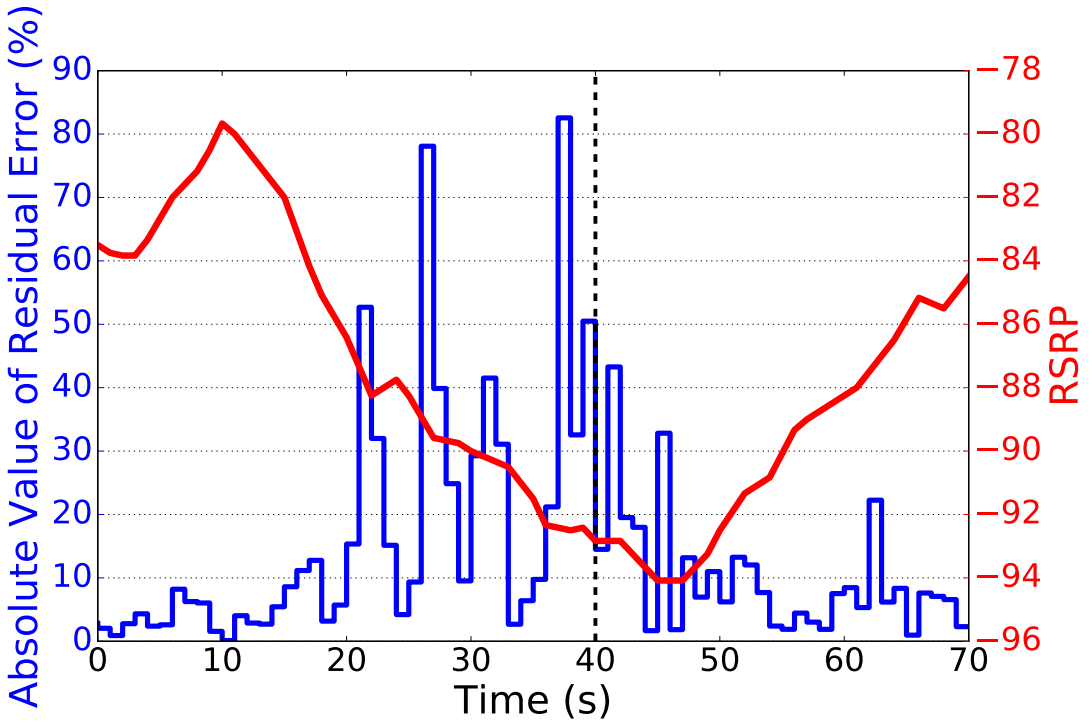


Figure 5.14: ARE vs. RSRP

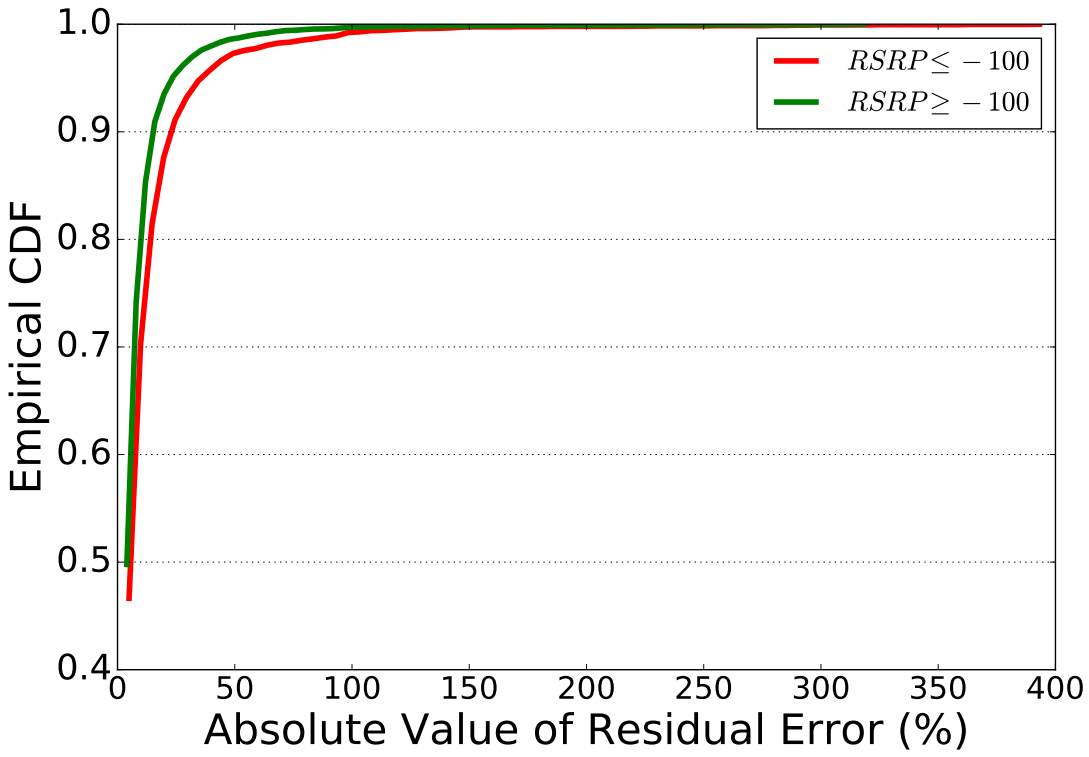


Figure 5.15: Empirical Cumulative Distribution of ARE for two RSRP ranges

model could identify devices approaching the edge region more accurately. Also adding signal-related information for neighbouring cells could help in decreasing errors in the edge region of the cell.

5.3 Video experiments in a controlled environment

This section evaluates the impact of improved throughput estimation on the streaming performance using a lab testbed. The testbed described in Section 3.2.1 is used. Next, the setup is presented, followed by streaming performance results.

For evaluation, three algorithms are selected: ARBITER+, BOLA-E, and EXO, with more information about each algorithm outlined in Section 3.2.3.2. Selected algorithms use information from both bandwidth estimators, as well as from buffer occupancy when deciding on the rate of the next chunk. These three algorithms use different bandwidth estimation techniques (EWMA, average and median). These techniques represent different approaches in bandwidth estimation, ranging from more conservative estimate (median) to more aggressive (EWMA and average).

For buffer length, the recommended values for each algorithm are used (60-seconds for ARBITER+, 32 seconds for BOLA-E, and 30-seconds for EXO). The initial delay is set to two chunks. After a stall event, play is resumed after one chunk finishes downloading.

5.3.1 Throughput prediction module

Experiments are based on a subset of the 4G dataset traces to illustrate the benefit of prediction. The same reasoning for trace selection is used as in Section 4.3.1. For characterisation of selected traces, standard deviation of throughput within a trace is considered. We sort traces based on standard deviation of bandwidth. Table 5.4 shows throughput statistics for the top 20% and bottom 20% of sorted traces.

Majority of traces with high bandwidth variability are collected in the highly mobile environment (car), while traces with low bandwidth variability were collected while devices were static or moving at low velocity (pedestrian).

Table 5.4: Throughput for selected traces

Type	Avg (Kbps)	Std (min - max, Kbps)
Low-variable traces	1656	591 - 1166
High-variable traces	4451	4010 - 6447

The throughput prediction module is implemented using the proposed prediction approach described in Chapter 5. When testing the streaming performance using one of the selected traces, this trace is eliminated from the training set of the prediction engine used in this experiment. The prediction engine is then used to identify the predicted throughput for every record in the trace. The predicted throughput is stored offline and provided to the video client during the experimentation. In summary, the tested trace is removed from the training procedure to ensure unbiased prediction values.

The presented prediction setup is close to reality as it does not imply the knowledge of the transportation mode (e.g. static, pedestrian, car etc.). Note that the prediction engine is based on traces with mixed mobility patterns. History is set to 20 seconds and different horizon values from 12 seconds-32 seconds are tested. The choice of horizon values was driven by results in Chapter 4, indicating that longer horizons benefit HAS players more than shorter horizons.

Figure 5.16 shows ARE across five different prediction horizons with history set at 20 seconds. Mixing mobility patterns does not change the performance trends established in Section 5.2.6. The prediction accuracy increases with the horizon. For the 12-second horizon, 90^{th} percentile ARE is less than 16%. Furthermore, this error drops below 10% for the longest, 32-second horizon.

5.3.2 Integrating predicted throughput in HAS algorithm

The predicted throughput can be integrated into the adaptation logic in two different ways. First, the predicted throughput may replace the entire throughput *estimation* in the algorithm (*E-type*). Alternatively, the predicted throughput may be used to replace the estimated throughput *samples* (*S-type*). Chapter 4 shows that for ideal prediction, algorithms with more conservative bandwidth estimation (harmonic, median) have higher improvement with direct estimate while algorithms with more aggressive bandwidth estimation (EWMA) prefer feeding prediction as a sample. As a result, for the EXO and BOLA-E direct estimate is used for prediction value, while for the ARBITER+ values are fed through bandwidth estimation module.

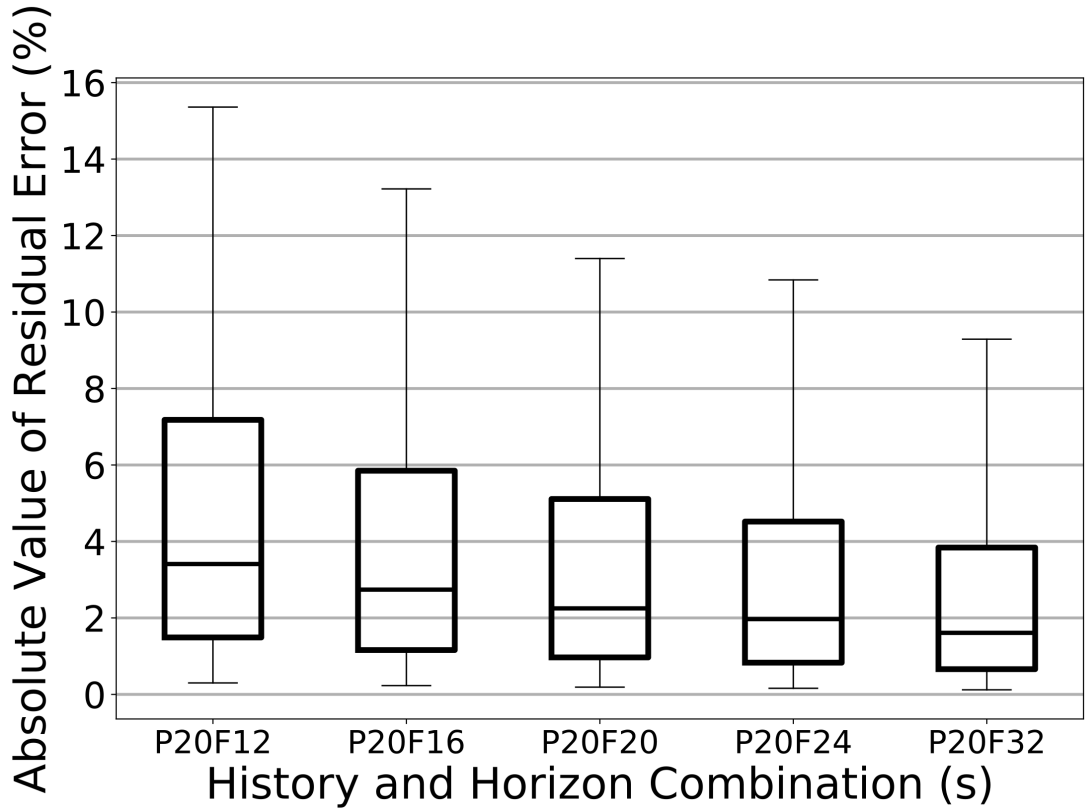


Figure 5.16: ARE accuracy for five different prediction horizons (mixed-mobility ML model)

5.3.3 Video QoE models

To evaluate the performance of HAS algorithms, standardised QoE metrics are analysed, such as average video bitrate, switching behaviour (e.g. stability), stall frequency and duration. In addition two video QoE models are used, *Yao* and *Clay*. The definition and main characteristics of these metrics and models can be found in Section 3.2.3.4.

To represent an overall QoE score, the geometric mean of the two QoE models is calculated. Both models are derived based on subjective tests, making it challenging to select only one. The geometric mean is chosen instead of the arithmetic mean because model scores are on a different scale (Clay 0-5, and Yao 0 - 100). Table 5.5 summarises QoE metrics and their notation.

Table 5.5: QoE Metrics Notation

Metric	Summary
Bitrate	Average bitrate
Stability	Average stability, equal to $1 - i$ with i being instability as defined in [JSZ14]
$Stalls_{num}$	Average number of stalls
$Stalls_{dur}$	Average stall duration
QoE	Geometric mean of Clay and Yao QoE

5.3.4 Streaming performance results

Streaming performance is evaluated without prediction as a base case and with prediction using different horizon durations. Evaluation is repeated for two scenarios, including 4-second and 8-second chunk duration (typically chunk duration is 2-10 seconds [BBHZ19]). The shown performance results represent the average of 10 runs (results are consistent even with five runs).

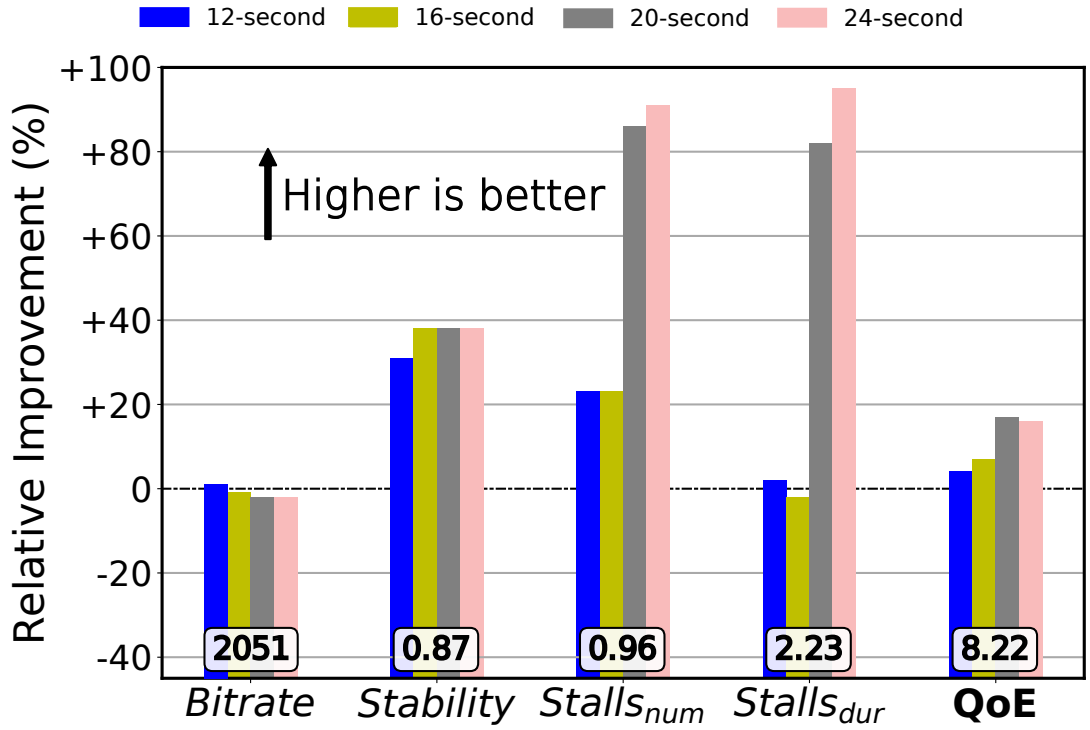


Figure 5.17: ARBITER+: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

For the 4-second scenario, Figure 5.17, 5.18, and 5.19 plot the relative improve-

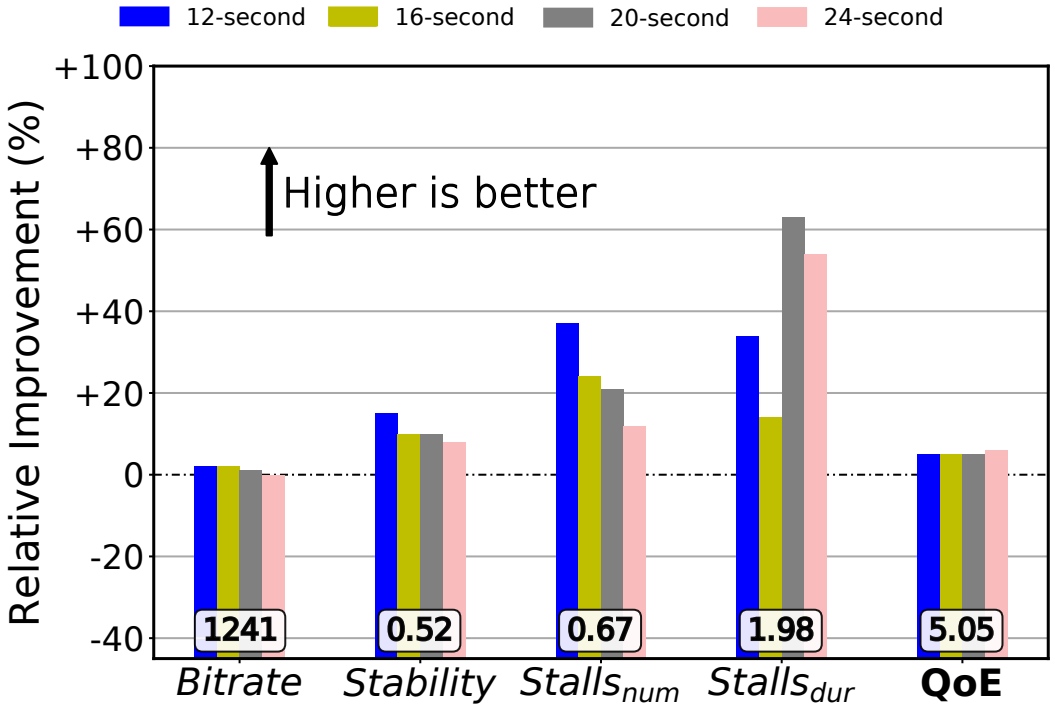


Figure 5.18: BOLA-E: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

ment in performance metrics, for the evaluated algorithms, relative to the no prediction case. Hence, a larger relative improvement in the streaming bitrate means a higher rate while a larger relative improvement in the number of stalls means fewer stalls. The performance metrics of the no prediction case for every algorithm are shown in the white boxes just above the x-axis. Figures show that integrating our prediction noticeably improves the QoE metrics of different algorithms. Specifically, prediction enables all algorithms to reduce/eliminate stalls. Additionally, prediction allows HAS algorithms to improve switching stability. In particular, average stability can be improved by 15%-47%. Furthermore, improving the accuracy of bandwidth estimation enables the algorithm to enhance their selected chunk quality. For example, integrating prediction fixes throughput underestimation with EXO leading to a higher chunk quality (by 16%), Figure 5.19. On the other hand, incorporating the prediction with ARBITER+ results in a negligible lower average chunk quality (2%), Figure 5.17. All these improvements add up leading to boosting the overall user QoE by 8%-27% in the 4-second chunk scenario. It is evident that BOLA-E is the least beneficiary of the compared algorithms, Figure 5.18. However, this is expected due to its design relies on a buffer level as the main quality selection decision and only uses throughput estimates

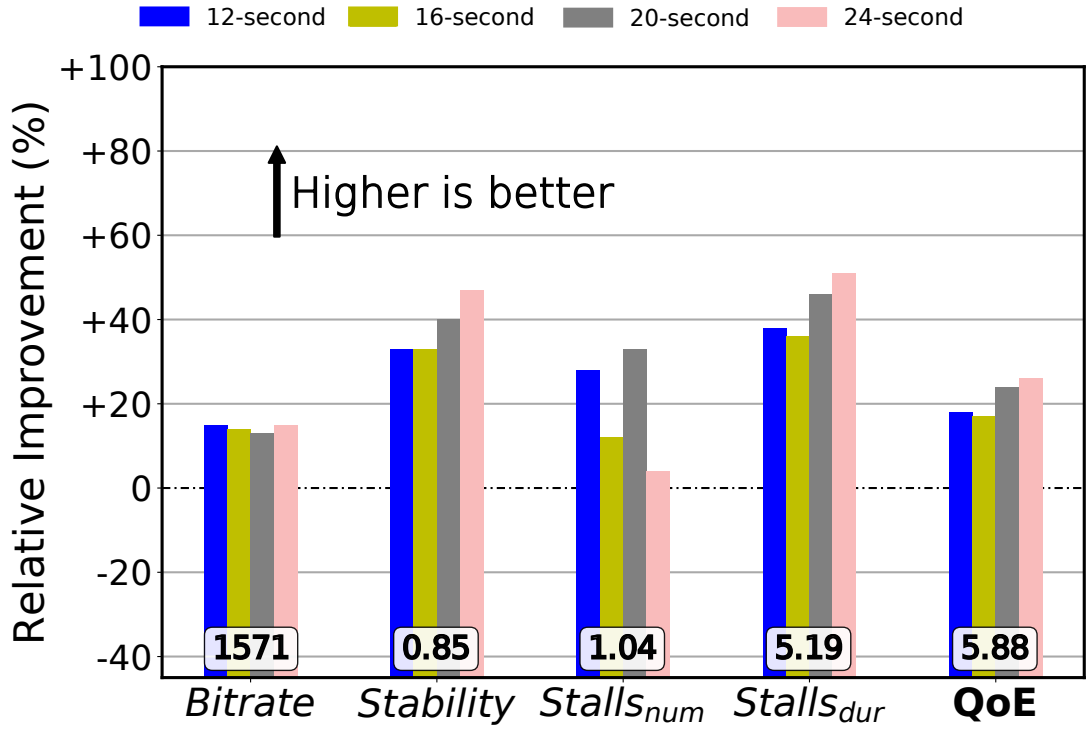


Figure 5.19: EXO: Relative improvement of different QoE metrics for the 4-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

in very limited cases as illustrated in Section 3.2.3.2.

In the 8-second scenario, Figure 5.20, 5.21, and 5.22 depict the relative improvement of the performance metrics for the evaluated algorithms. Similar to the 4-second case, the prediction shows a positive impact on all metrics in the majority of the traces. Overall, QoE can be improved by 19%, 13% and 33% for ARBITER+, BOLA-E, and EXO, respectively. This improvement is higher than that attained in the 4-second scenario. Fewer opportunities to change the quality and react to sudden changes to channel capacity in the longer chunk case explains this behaviour. In both 4-second and 8-second scenario, EXO features the highest relative improvement in QoE score. This improvement is attributed to the increase in the average bitrate (15% improvement compared to 1-5% for ARBITER+ and BOLA-E).

Identifying the optimal horizon duration is an essential design parameter. For the 4-second scenario, the algorithms show a similar QoE improvement for 20-24 second horizon. In the 8-second chunk duration scenario, algorithms show distinct performance as the prediction horizon increases. ARBITER+ shows the best QoE relative improvement with a 32-second horizon, while BOLA-E and EXO

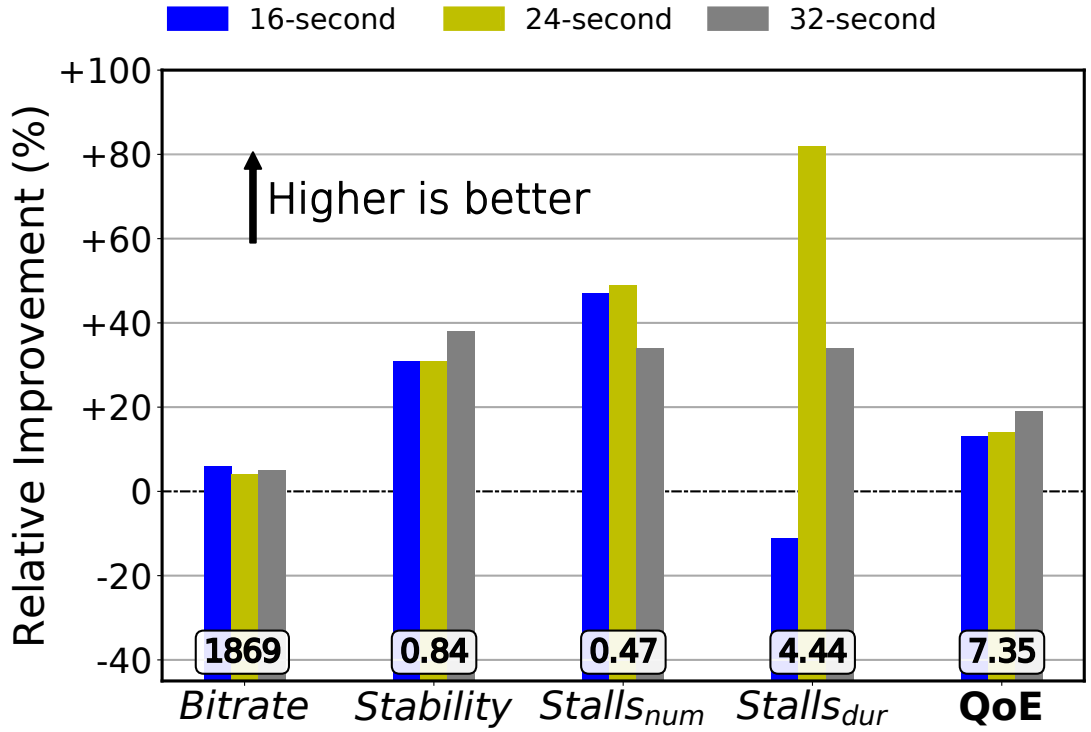


Figure 5.20: ARBITER+: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

achieve the best performance with a 24-second horizon. Extending the horizon results in averaging over a longer period and thus reducing variability between subsequent prediction values. This leads to an improved switching performance. Additionally, longer horizon enables a client to proactively switch quality and avoid stalls when the throughput drops for a relatively long time that can deplete the buffer. However, increasing horizon will eventually lead to client inability to adapt to sudden changes in channel capacity. This can be seen in case of 8-second chunk and EXO algorithm, Figure 5.22. For the 32-second horizon, stability improves by 33%. However, this stability leads to a decrease in stall performance (compared to other horizons) and a sharp drop in overall QoE.

Finally, there is a discrepancy between improvement in each QoE metric and overall QoE improvement. This is a direct consequence of QoE model we use in this study. To understand this behaviour, each QoE model is analysed individually. Let's analyse particular trace where we have high improvement in rebuffering performance (ARBITER+, 4-second chunk duration). Reduction in a number of stalls and total stall duration results in 35x and 1.9x higher stall impairment for Yao and Clay, respectively. For switching, impairment is 1.6x higher for Clay,

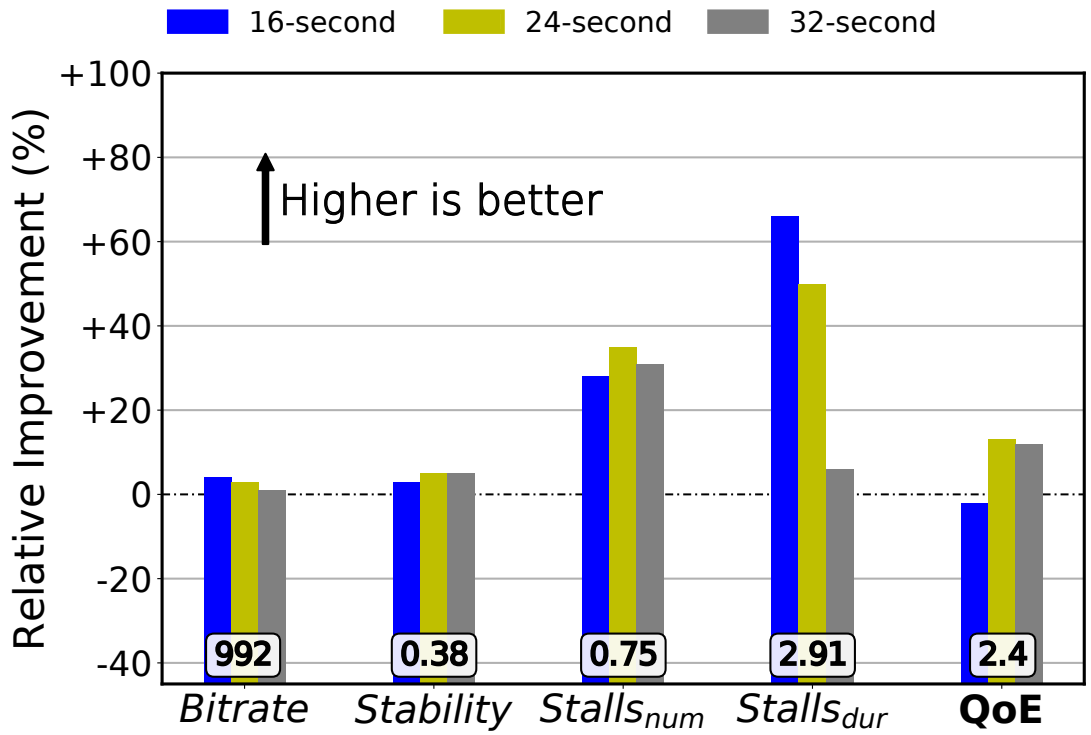


Figure 5.21: BOLA-E: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

compared to 1.25x for Yao. While Clay produces similar tradeoff between stall and switching impairment, Yao gives much higher weight on the stall reduction. As a result, the prediction improves Yao QoE by 40%, while for the Clay, lowers QoE by 44%. Overall, QoE improves by 14%. A similar observation holds for BOLA-E.

5.4 Real-Time prediction

Motivated by results of the lab experiments, prediction engine is implemented inside mobile devices leveraging the Android API and an existing Java ML library. For the ML library, Weka[2] is used, a software framework that has an extensive collection of state-of-art machine learning algorithms implemented in Java. While not explicitly designed to run on mobile devices, it represents a good starting point for testing the initial prototype.

For the collection of radio metrics and device velocity, we use classes and methods

[2]https://www.cs.waikato.ac.nz/ml/weka/

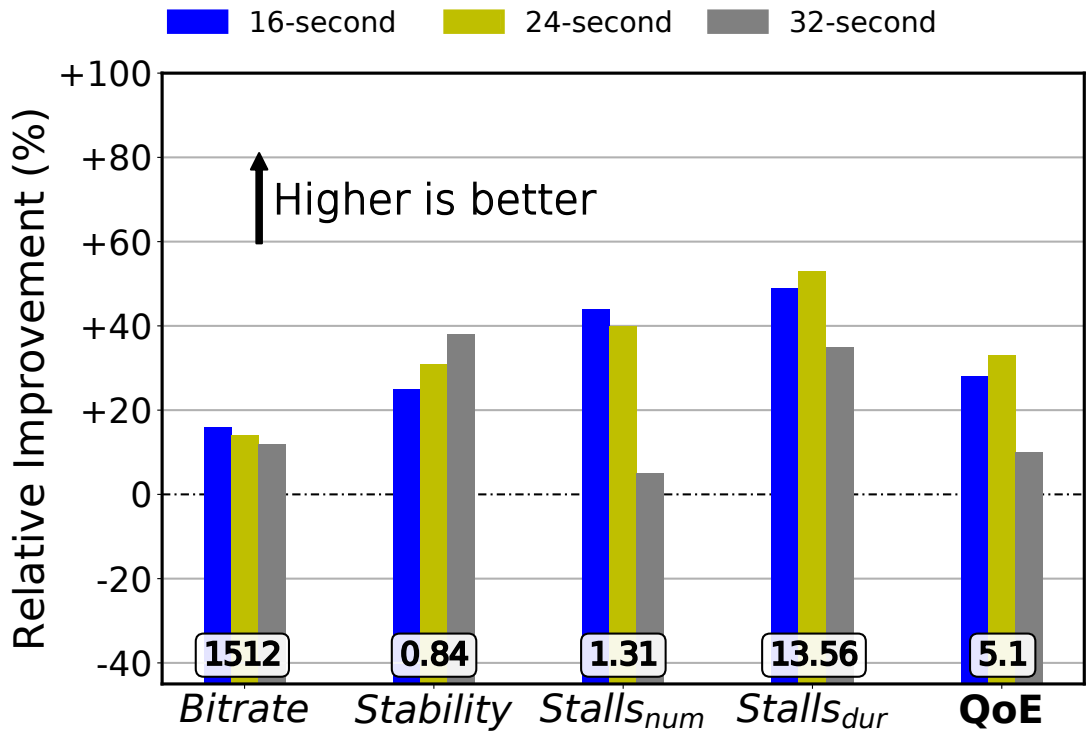


Figure 5.22: EXO: Relative improvement of different QoE metrics for the 8-second chunk duration (*The metrics are normalised to the no-prediction case with numbers in white boxes representing the value of each metric for the no-prediction case*)

detailed in Table 5.6. Values are collected periodically, every 1-second in a separate thread inside ExoPlayer. All metrics are stored in First In First Out (FIFO) queues with size limited to 20 values per queue. ML model (Random Forest) is trained offline and ported to the mobile devices. For the HAS algorithm, EXO is selected using the same parameters as outlined in Section 5.3. 20-second prediction horizon (direct estimate) is used. The prediction value is generated in the following way: every time a decision for the next quality needs to be made, the adaptation logic requests a prediction value. This value is generated by creating Quantile statistics based on current state of FIFO queues, followed by a call to the model itself with statistics as input. Finally, the model returns a prediction value for the next 20 seconds.

56 static and mobile field tests are performed in a real cellular network. Each experiment consists of two mobile devices (same model) streaming the same video content side by side. One mobile device stream content with no throughput guidance, while the other one uses our throughput prediction as outlined above. To minimise non-radio related effects, all tests are performed in the early morning while assuming the network is not busy. The following sections explain the device

Table 5.6: Android API classes used for collecting radio and throughput metrics

Metrics	**Class/Method**
RSRQ	*CellInfoLte/getCellSignalStrength().getRsrq()*
RSRP	*CellInfoLte/getCellSignalStrength().getDbm()*
CQI	*CellInfoLte/getCellSignalStrength().getCqi()*
SNR	*CellInfoLte/getCellSignalStrength().getRssnr()*
Velocity	*LocationManager*
Throughput	*TrafficStats*

and model limitations faced while implementing the prediction engine in a mobile device.

Device Limitation: Standard Android library is leveraged for capturing channel metrics. However, implementation of these callback functions depends on the manufacturer of the mobile System on Chip (SoC) chipsets. Also, not all parameters are reported for different cellular technologies (2G/3G/4G). We use Samsung J5 mobile devices, as the Exynos chipset implements almost all Android callback methods for reporting channel metrics. Furthermore, loading and running ML model inside a mobile device can be challenging. Due to hardware limitations, loading of the appropriate model can take up to several minutes. e.g. training Random Forest model on all traces results in a large model ($400MB$) which can not be loaded in the mobile device. As a tradeoff, the number of trees is limited to 30 and tree depth to generate a smaller model. This tradeoff results in accuracy decrease of the model. As a result, prediction error (90^{th} percentile) increases from 13% to 21%.

Experiment Limitation: Running two devices side-by-side at the same time does not necessarily mean same channel and environment conditions. There are a couple of limitations we faced during these trials. First, devices do not necessarily report the same values for metrics. Second, even in the static case, phones can be connected to different evolved Node Bs (eNodeBs) or same eNodeB but different cell sector. While effort is done to minimise these occurrences, there is little or no control over this in mobile cases.

Figure 5.23 shows similar trend for QoE metrics as in Section 5.3. However, there exists one major difference between experiments performed in the previous section and experiments in a real cellular network. While we selected a certain subset of traces in controlled experiments (in particular, traces with an average rate close to highest video rate), in a real environment, the majority of tests were conducted in a high channel capacity environment (with the average rate

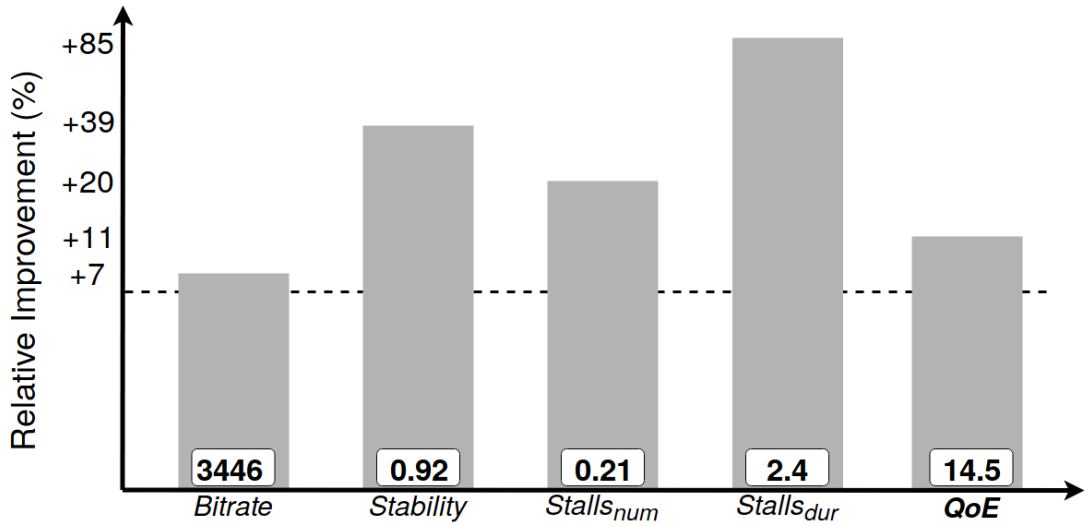


Figure 5.23: Relative improvement of different QoE metrics in real cellular network with respect to the no-prediction case (EXO algorithm, 4s chunks)

greater than 6Mbps). This is most evident in the case of bitrates. Average bitrate across all session is 3.4Mbps (compared to 1.5Mbps in a controlled environment). As a result, the impact of prediction is limited as the highest rate is 4.3Mbps, leaving less space for improvement. Still, prediction improves all QoE metrics. In particular, improvement in bitrate is only 7%, capping QoE improvement to 11%.

5.5 Discussion

Arising from this chapter, a number of matters arose which merit further discussion and research, which are considered here.

5.5.1 Higher data granularity

Additional analysis is performed of the summarization techniques concerning higher measurement metrics granularity than is available using the Android API. For these, a 250ms granularity is obtained from a Qualcomm Diagnostic Tool, (QXDM)[3] a tool capable of capturing device-level metrics at high resolution (hundreds of ms) directly from the hardware. QXDM is a properiaty tool working only with Qualcomm chipsets and thus not universally applicable. Over sixty traces

[3] `https://www.qualcomm.com/`

are collected for different mobility patterns with an average duration of 15 minutes per trace. The following results are based on a mobile scenario. For 250ms measurements granularity, higher sampling frequency results in better overall prediction accuracy. Figure 5.24 depicts ARE for different combinations of history and horizon values for 1s and 250ms data granularity. Regardless of the horizon, higher measurement granularity results in massive improvement for ARE (compared to one-second granularity), with the 90^{th} percentile of ARE below 15% for all cases. Furthermore, average error 7%, 5.5%, and 5% for prediction horizons 4, 8, and 12 seconds respectively (with only four-second history interval). Also, a similar trend is observed as with one-second granularity, with prediction accuracy improving with increasing history and horizon lengths. In all cases higher data granularity significantly improves prediction accuracy compared to 1-second setup.

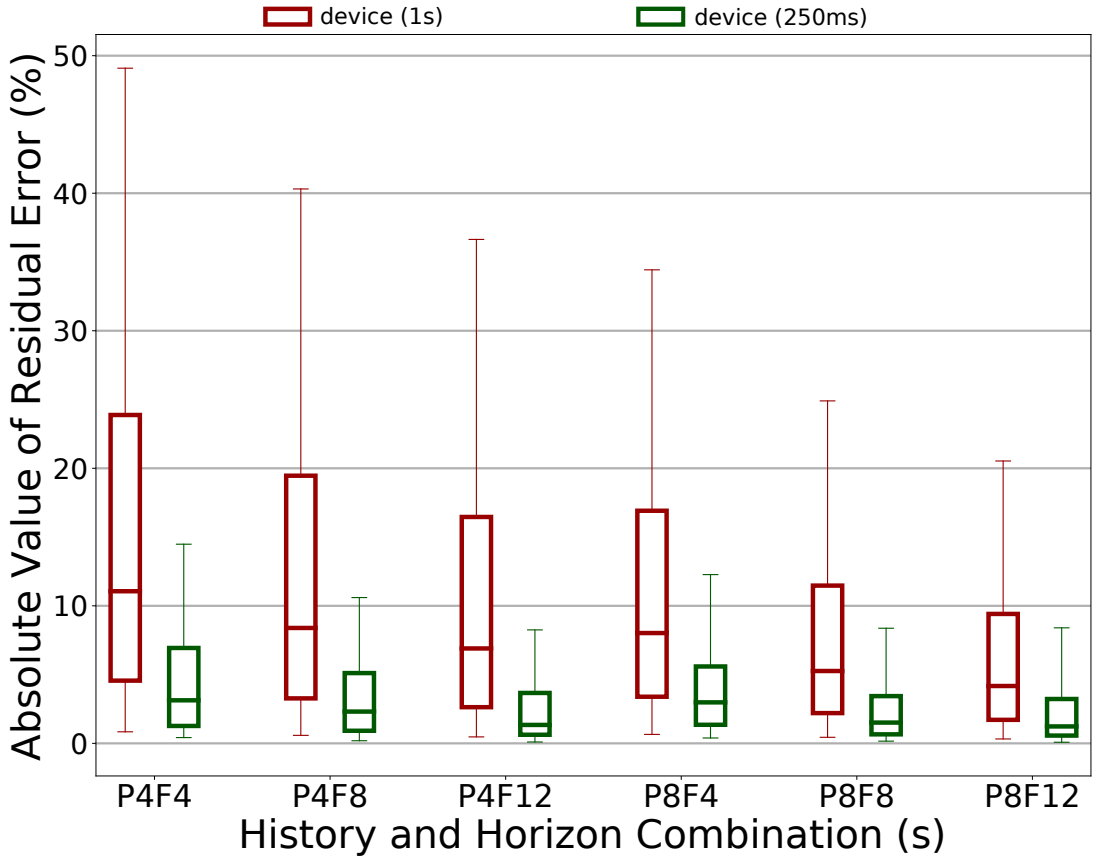


Figure 5.24: Comparison of ARE for fine-grained data for 250ms and 1s granularity (quantile summarization technique)

5.5.2 Network data sources

As outlined in Section 5.2.1, device-level, and network-level metrics represent two types of measurement data. Device-level metrics represents a device channel, context, and throughput values. However, the predictive power of device-level metrics is limited, as the device cannot infer the state of other devices that are sharing network resources with this device. For example, during busy hours, prediction values may overestimate throughput capacity as a correlation between user channel conditions and available throughput deteriorate. To alleviate this issue, network-level metrics are needed, e.g. cell load would indicate how busy cell is, preventing overestimation and increased probability of rebuffering events. Also, network operators may inflict throttling in a network [KLC+16] during busy hours which can skew prediction system completely. Additional limited analyses is performed including network-level metrics in throughput prediction-related experiments. While network-level metrics provide significant improvement in throughput prediction accuracy, their use in real-time is still open question and requires further exploration.

For network-level metrics, the following measurements are considered:

- Competing throughput - average throughput of the devices connected to a given cell.
- Competing CQI, RSRP, RSRQ and SNR - average per metric value of all devices connected to the same cell
- Load - number of devices connected to the same cell and Physical resource Block (PRB) utilisation

On representing competing device metrics, the average value is used across all devices as the number of users per cell dynamically varies with system load.

Network-level metrics are collected by a set of instrumented cells to which the phone and laptop were connected. For a given device, a cell is instrumented to log its network-level metrics. Certain metrics, e.g. cell site load and PRB utilisation, are reported with a fixed periodicity and other 'session level metrics', e.g. CQI, are reported for an entire session whenever the Long-Term Evolution (LTE) bearer tears down. To tear down a bearer, one simply needs to idle a device activity for a few seconds. Accordingly, devices are instrumented to initiate a download (*active* period) and then pause for a few seconds to cause a bearer tear down event (*idle* period). Selecting the length of active periods requires balancing

among competing concerns. Because only one value is collected of any 'session level metric' for the entire active period, the active periods should not be too long. At the same time, prediction horizons of reasonable lengths are needed. But to train and test prediction horizon of length x seconds requires collection throughput measurements from *contiguous* intervals of length at least x seconds. As a compromise, active periods of 16 seconds are used.

Figure 5.25 illustrates this difference. For the same combination of history and horizon interval, network-aided prediction improves prediction accuracy significantly, compared to the device-only approach. Also, these results illustrate one important characteristic of having comprehensive information about the finite resource available. Even with few historical data, a network-aided approach achieves higher prediction accuracy. In contrast, a device-only approach needs a longer history to indirectly infer its environment and achieve similar prediction accuracy as a network-aided approach.

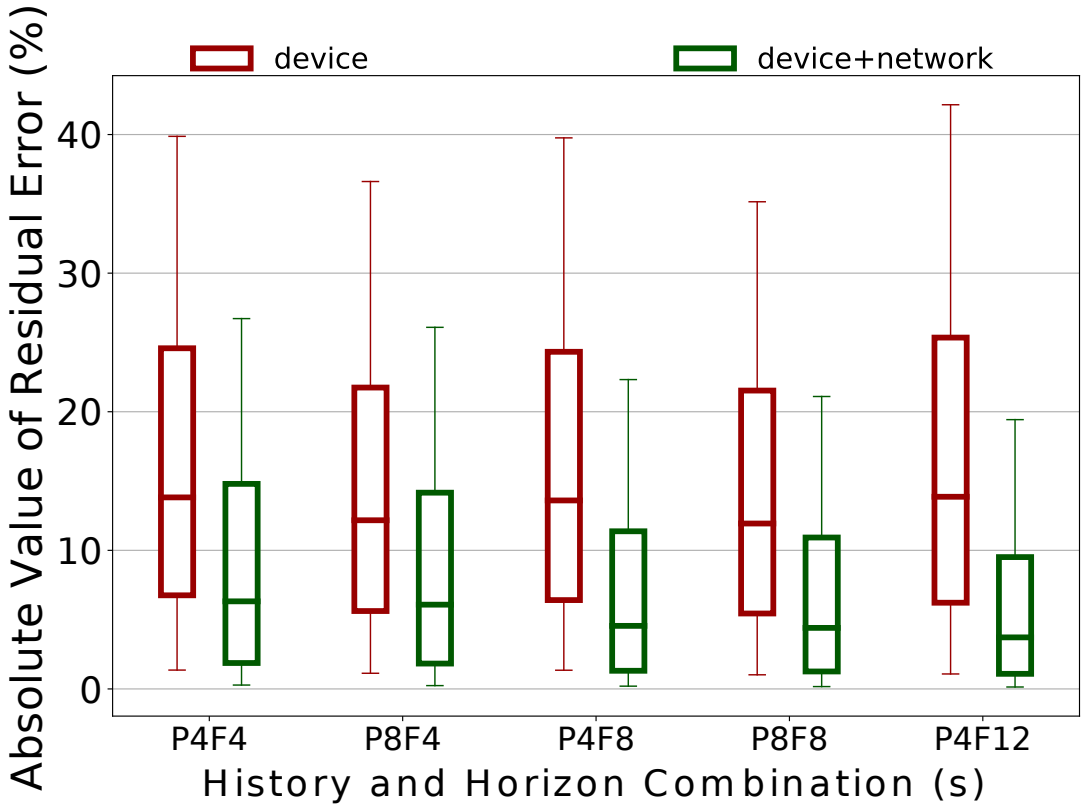


Figure 5.25: Comparison of ARE for device and device+network approach (aperiodic sampling interval)

5.5.3 Deriving optimal high-level features

Deep learning frameworks (TensorFlow Lite for Android and Core ML for Apple iOS) are optimised to run on mobile devices. Deep learning architectures may help in obtaining even more accurate predictions. In Section 5.2 the presented solution represents history with multiple measures of variability helps drive down prediction error. However, standard statistical measures are used, which are not necessarily the optimal ones. Also, the same statistics are used across all metrics. However, optimal statistics for one metric may not necessarily be optimal for other metrics. Instead of "handpicking" high-level features from *raw* input, using neural networks with multiple layers can automate extraction of more significant features from input data. This observation leads to deep learning algorithms, and in particular recurrent neural networks. Sequence-to-sequence neural networks [SVL14] have a broad reach in language translation. The main power of these networks lies in the ability to summarise a sentence from one language by a couple of critical values (encoder) which then can be used to translate it into another language (decoder). Similar architecture can be employed for throughput prediction, where the history of each metric will be summarised by optimal measures of variability independently of each other. To motivate further research, Long Short-Term Memory (LSTM) model is trained. LSTM represents a family of Recurrent Neural Networks (RNN), more specifically gated RNNs. RNN represent a type of neural networks design to process sequence of data. RNN networks are a more suitable choice over other neural networks for time-based tasks, such as throughput prediction [ZPH19].

The ARE accuracy of LSTM is compared against RF and Support Vector Machines (SVM). SVM is based on constructing hyperplanes for making decision boundaries that separate between points of different classes (e.g. for two class case, hyperplane would represent a line separating them). For making separation easier, input features can be transformed by appropriate functions called kernels [MSBZ10]. For SVM, parameters tuning represents the main drawback. A common technique, grid search, can be used for automation of search process in finding optimal values for algorithm parameters. However, grid search is time-consuming.

Figure 5.26 shows the boxplot of ARE for the *raw* input and *quantile* summarization technique for different ML algorithms. These results are based on device-level data collected at one second granularity for P20F20.

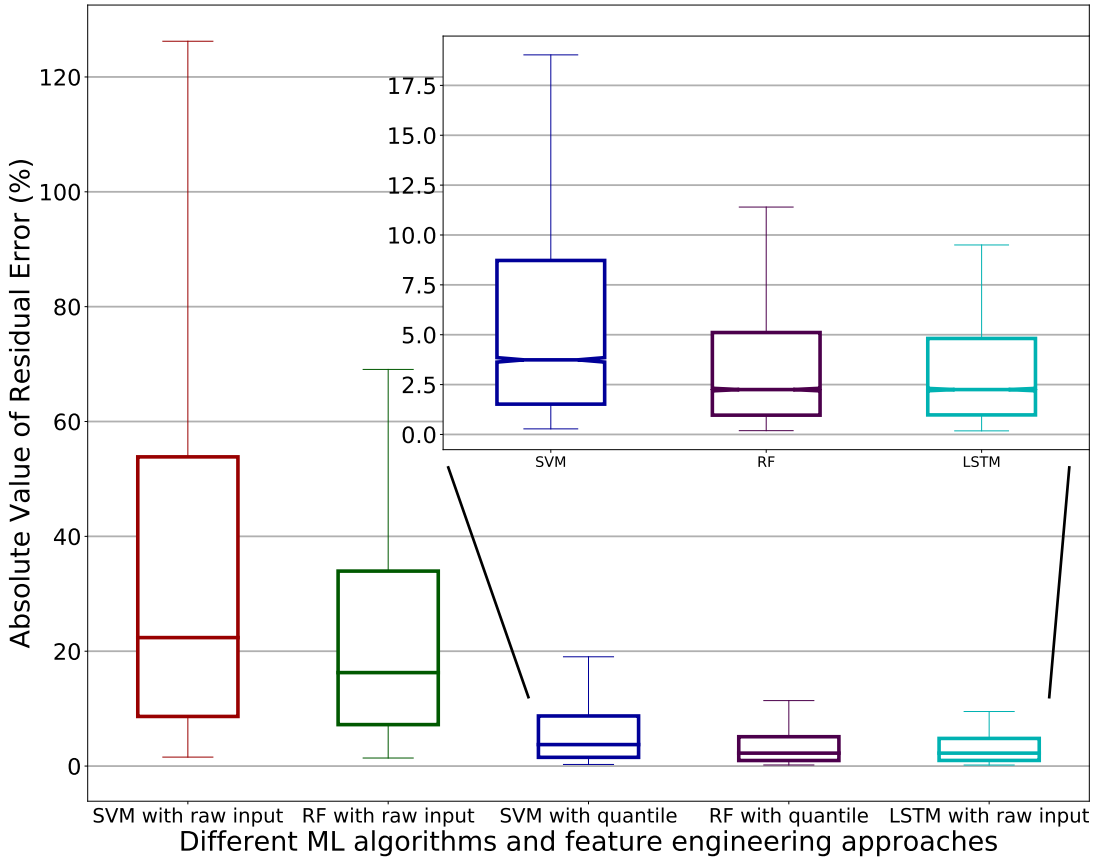


Figure 5.26: Comparison between different data representation approaches and ML algorithms for 1-second data granularity

Results confirm the importance of extracting high-level features from *raw* input as SVM yield significant improvement in throughput prediction accuracy. Additionally, LSTM achieves similar results operating directly on *raw* input. Furthermore, LSTM achieves lowest 90^{th} percentile of ARE among all algorithms, indicating a better learning ability for rare patterns.

5.6 Conclusion

It is known that the cellular radio access network offers highly variable performance due to a variety of factors. This chapter addressed the problem faced by applications such as video streaming in trying to estimate the available throughput over the cellular network. Prior work has focused on the use of a small set of performance metrics gathered by an end-user device to make predictions up to one second. Machine learning approach is used in this chapter to leverage a broad range of cellular measurement metrics to make predictions several seconds

into the future. A thorough quantitative study of throughput prediction in a real cellular network is conducted. Combining machine learning techniques with radio channel metrics summarised by a novel quantile abstraction technique achieves low throughput prediction errors (90% of errors below 18%). By utilising the proposed abstraction technique, the prediction engine is able to capture trends and variation in metrics data accurately, even in the environment where metrics are updated/available at low time granularity. Furthermore, the benefit of adding network-related information about the cell environment is quantified, showing that it can decrease error further by 21% in some instances. The relationship between the length of historical data and the future prediction accuracy is assessed, showing that increasing history length of measured metrics helps in improving prediction accuracy by 50% on average and that increasing future horizon has a similar effect, resulting in an increase of prediction accuracy by 16%. Work outlined in this chapter provides a convincing case for further development and deployment of cellular network prediction based on machine learning techniques.

The impact of throughput prediction generated by the ML prediction model is investigated with state-of-the-art HAS algorithms on overall user QoE. Prediction engine was implemented on a real mobile device, and additional experiments are performed in a real operational cellular network. Having more accurate predictions allows three adaptation algorithms to improve its performances. All tested algorithms improve all QoE metrics when using prediction. Notably, prediction reduces stalls by up to 85%, and bitrate switching by up to 40%, while maintaining or improving video quality. As a result, QoE score improves significantly, by up to 33%. Furthermore, experiments performed in a real operational network has enabled identification of challenges that motivate further research and will be of interest to practitioners seeking to implement ML-based prediction in real systems.

This chapter completes the analysis of throughput predictions in a cellular network, from the motivation for need for throughput prediction (Chapter 4) to ways to obtain accurate predictions by leveraging physical channel information, and finally quantifying the impact of throughput prediction on HAS performance (this chapter).

Chapter 6

Impact of Network Queues on video performance

With the ongoing growth of video traffic on the Internet, conveyed through HTTP adaptive streaming (HAS), specific concerns emerge regarding performance related interactions between video and non-video traffic. The bandwidth demanding nature of HAS traffic can cause the well-known *bufferbloat* phenomenon harming the performance of delay-sensitive traffic such as web and VoIP. This chapter investigates whether it is feasible to manage the performance interactions between HAS and non-HAS traffic through changes to the network operation, specifically the configuration of network queuing, so as to improve the overall quality of experience.

Previous studies of network queuing have examined conventional HAS traffic. However, HAS traffic is a relatively new traffic type whose impact on other traffic types, and vice versa, is largely unknown. Additionally, it exhibits characteristics of both long-lived flows (spending most of the time in HAS congestion avoidance phase) and short-lived flows (frequently going through the Transmission Control Protocol (TCP) slow-start phase). Furthermore, compressed video is typically Variable Bit-Rate (VBR) in nature. Hence, HAS operates at a discrete set of rates that vary due to both adaptation decisions and VBR encoding. Thus, relying on prior studies of non-HAS TCP is inappropriate, and comprehensive research is needed to better understand the application and traffic dynamics when HAS and non-HAS compete. Of note is the work by Hong et al. [HMW15] who investigates using network simulations, measuring fairness and utilisation when multiple traffic types share a bottleneck employing Active Queue Management

(AQM). However, using these objective metrics it is not possible to determine the effect on user Quality of Experience (QoE), which for video, in particular, is the accepted metric since it also captures changes in quality. The work in this chapter is distinguished by its focus on using real experiments and traffic in understanding of the impact of network queuing design (including modern AQM techniques), and measuring the impact on application-level metrics. Specifically, the interplay between content, network, and application in various scenarios is analysed, while seeking to validate the following key hypotheses:

1. The well-known 1xBandwidth Delay Product (BDP) rule-of-thumb for dimensioning network queues can achieve full link utilisation for multiple HAS video clients.

2. Increasing router queue length will improve fairness among competing HAS video clients with heterogeneous round-trip-times (RTTs).

3. Use of AQM techniques will protect web client performance (in particular page-loading time) when sharing resources with HAS video clients.

The first hypothesis seeks to confirm previous studies that argue that full link utilisation is achievable even with a fraction of BDP. The second hypothesis leans on TCP rate being inversely proportional to round-trip time. Flows with shorter RTT will fill the queue more frequently and with a larger queue; those packets will experience larger queuing delay lowering overall rate and improving overall fairness. Finally, modern AQM techniques are designed to protect delay sensitive traffic regardless of competing traffic characteristics. They track queue delay and randomly drop packets keeping queuing delay low while maintaining high link utilisation.

The research is guided by validation of these three hypotheses, and presents a realistic experimental study that provides a definite answer to the research question posed above. This empirical analysis sheds light on the interaction between content (e.g. video rate), link (e.g. capacity, round-trip time) and queue design. Our findings are based on real traffic generated from video players (using well known HAS adaptation algorithms) and a popular web browser. Interaction between these different types of traffic is enhanced with a realistic user behavioural model for web traffic. Finally, the last chapter introduces a two-queue scheduling discipline that solves challenges raised by previous hypotheses.

The systematic empirical study performed in this chapter yields several important practical results.

1. The well-known "rule-of-thumb" (1xBDP) for network queue dimensioning causes underutilisation (70%) when multiple HAS clients share a bottleneck.
2. Larger queues, e.g. 2xBDP, can improve utilisation and quality by 15% on average.
3. Larger queues also help in improving system fairness for clients with heterogeneous RTTs. However, larger queues can negatively impact delay-sensitive traffic, causing the *bufferbloat* phenomenon.
4. Replacing First In First Out (FIFO) with AQM does not improve Page Loading Time (PLT) significantly, in which AQM is unable to counter the bandwidth-hungry nature of HAS flows.

6.1 Methodology

First, the effects of queue size on HAS performance is investigated. Queue size is defined as a product of link capacity and RTT. The next compelling step includes changing values of these factors itself, quantifying their impact on overall HAS performance. The main focus is on the multi-client environment, so analysis of a different number of clients presents a logical stride especially because of video clients inherent ability to adapt to available network resources. FIFO queue is then replaced with AQM disciplines and web traffic clients are introduced in efforts to answer the question about AQM ability to keep a low delay for delay-sensitive applications. AQM techniques are tested with different combinations of video and web clients quantifying the impact of HAS adaptation algorithms on web delay.

6.1.1 System model

In the experiments, the testbed framework for wired experiments is used as described in Section 3.2.2. N clients are considered sharing a bottleneck link and requesting their data from a remote HyperText Transfer Protocol (HTTP) server. Both video and web content are stored on the HTTP server. The role of the bottleneck link and network elements is to emulate scenarios with different queue size, queue scheduling techniques, link capacity, and RTT. Performance metrics for different applications are collected at the client side.

Table 6.1 shows default values and ranges for key parameters used in the experiments.

Table 6.1: Different Parameter values for experiments

Parameter Name	Default Value	Range
N	6	2-8
RTT (ms)	40	20-160
C (Mbps)	6	6-30
$Client\ Buffer(s)$	60,240	60-640
$Network\ Queue$	1xBDP	1-30xBDP

6.1.2 Outline of experiments

Experiments are split into two categories depending on the type of traffic in use. For the first part, video user QoE is analysed while varying the bottleneck link queue size, bottleneck link bandwidth, and the client-server RTT. The bottleneck link is configured using netem/tc [Kel]. In experiments, the bandwidth varies between 6-30Mbps and the RTT delay between 20-160ms. These values are typical for broadband connections in Europe [Eur13]. Next, FIFO is swapped with AQM queuing mechanism and experiments are repeated.

In the second part, the homogeneous traffic is replaced with a mixture of video and web traffic users competing for network resources. Experiments are repeated with FIFO and AQM queuing mechanisms and impact on video and web QoE is analysed.

For the AQM techniques, two disciplines are evaluated: Controlled Delay Management (CoDel) and Flow Queue (FQ)-CoDel.

For the TCP variant, TCP Reno (version NewReno defined in RFC 6582 [GHFN12]) is used as a TCP congestion algorithm. TCP Reno is a loss-based congestion algortihm [AABB19]. Compared to other TCP algorithms (Cubic and BBR), Reno shows similar performance [ALH+18].

Finally, both AQM and FIFO queuing disciplines are evaluated in the presence of heterogeneous RTT and its impact on video QoE is analysed. Figure 6.1 shows the roadmap of the experiments.

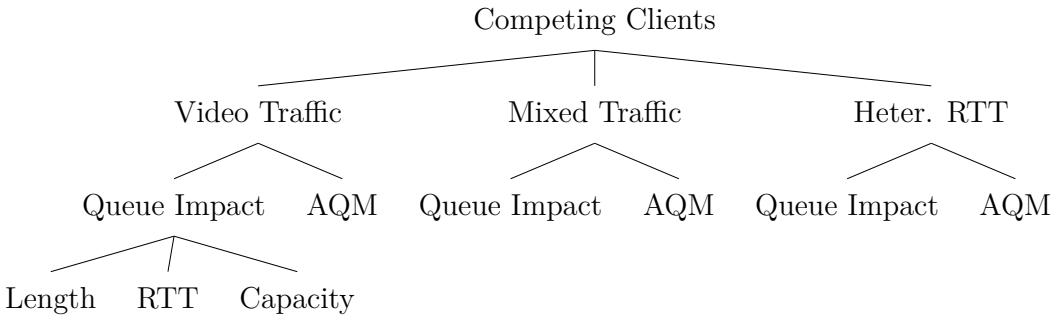


Figure 6.1: Summary of conducted experiments

6.1.2.1 HAS video client

As HAS client the well-known open-source player GPAC is used (see Section 3.2.2 for more details). For adaptation algorithms, FESTIVE [JSZ14], BBA-2 [HJM+14], and GPAC default algorithm are evaluated (Section 3.2.3.2 gives overview of characteristics of these algorithms). For all algorithms, 12 seconds of initial buffering and 4 seconds of rebuffering, when stalls occur, are considered. Different application buffer sizes are considered including 60-640 seconds. Typical buffer sizes observed in the literature vary from 30 sec [JSZ14] to 240 seconds [HJM+14]. While the 60 second buffer is a typical buffer size, the 640 second buffer is selected purely to allow observing the impact of eliminating "ON-OFF" behaviour on the streaming performance.

6.1.2.2 Video traffic

Eight clips (*An Idiot Abroad, Casino Royale, Harry Potter, Man of Steel, Star Trek, Avatar, Big Hero 6, and Star Wars*) are randomly selected (uniform distribution) from a HAS dataset described in Section 3.2.3.1. The segment duration for all clips is 4 seconds, which again is commonly used by popular services. All the experiments last for the duration of clips, which are 16 minutes long, served from an APACHE[1] server.

6.1.2.3 Web traffic model

For the web traffic model, a behavioural model is used as described in Section 3.2.3.3 mimicking realistic user browsing behaviour. In experiments, number of web clients varies between one and six.

[1] `https://httpd.apache.org/`

6.1.2.4 Active queue management

Since its inception in 1993 with Random Early Detection (RED) [FJ93] as a first published AQM discipline, the primary goals of these disciplines were having high utilisation and low packet loss and delay. A certain type of traffic, e.g. web traffic, is delay-sensitive and its performance degrades with increased delay as a consequence of full router queues.

CoDel [NJ12] is an AQM discipline based on packet-sojourn time as the leading indicator of packet delay. Packet-sojourn time represents a time packet spends in the queue. Taking each packet delay results in link-rate independence. CoDel adopts two constants *target* and *interval*. *Target* value is a threshold value for determining the standing queue. If the sojourn time exceeds a threshold (target) for at least interval, then algorithm drops the packet (in dropping state, intervals are reduced inverse proportional to the square root of the number of drops). CoDel's main characteristic is that it needs no configuration, which was the main drawback of its predecessor, RED [Ada13].

Flow Queue CoDel is a variant of CoDel that includes Deficit Round Robin as a queue scheduler. Packets go to different queues based on a hashing function [SV96]. CoDel operates on each queue [HJMT+16].

6.1.3 Key performance metrics

Main HAS performance metrics include player instability (i) [JSZ14, AABD12], fairness (F-Index), bandwidth utilisation (bw_{util}), average quality representation rate (r_{avg}) and stall performance.

Performance of web clients is measured by capturing page-loading time and web QoE.

The results shown represent the average of five runs, with 95% confidence intervals. In every run, each client randomly requests a video clip from the server and starts at most 4 seconds (randomly selected with uniform distribution) after the previous client. For web clients, the user randomly selects web page and reads a page for some time (as explained in 3.2.3.3).

6.2 Experiments with video traffic

This section starts by analysing how queue length impacts the QoE performance of multiple video clients. As the queue length is dependent on multiple factors including RTT, link capacity, and the number of competing clients, the impact of these parameters is explored as well. However, only exciting results and observations are reported.

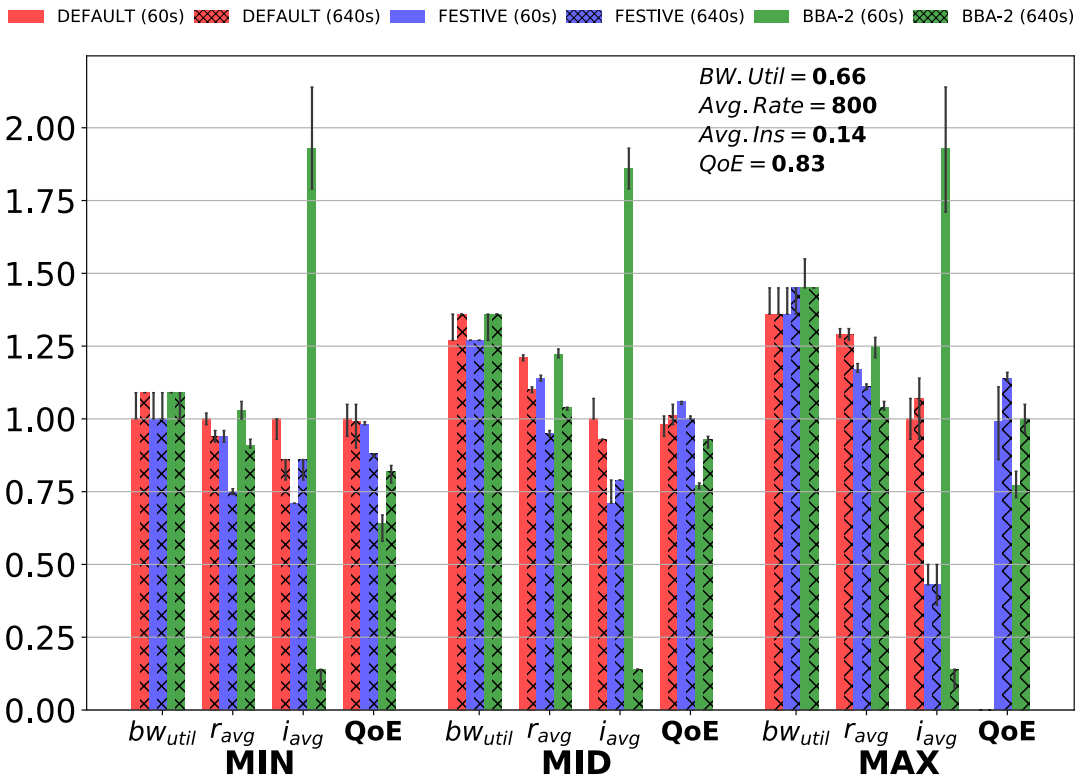


Figure 6.2: QoE metrics across different queue lengths

Table 6.2: Stall Performance across different queue lengths

Algorithm	Min	Mid	Max
Default 60s	(3,1,1)	(5,31,38)	(100,645,1348)
FESTIVE 60s	(0,0,0)	(0,0,0)	(13,20,28)
BBA-2 60s	(3,1,46)	(0,0,0)	(0,0,0)
Default 640s	(3,7,44)	(4,41,39)	(100,642,1239)
FESTIVE 640s	(0,0,0)	(0,0,0)	(0,0,0)
BBA-2 640s	(0,0,0)	(0,0,0)	(0,0,0)

Six video clients competing for a 6Mbps link with a 40ms RTT are considered. The size of the network queue varies from 1xBDP to 30xBDP. The 30xBDP is selected purely to allow observing the case where there would be no packet loss.

Figure 6.2 summarises key performance metrics for three algorithms with two buffer configurations (60 and 640 seconds) across different queue lengths. Queue sizes are grouped into three categories: *MIN* representing a 1xBDP case, *MID* 2-20xBDP (averaged), and *MAX* 30xBDP, respectively. Values are normalised against GPAC default algorithm with 60 second buffer and 1xBDP queue size. Similar, Table 6.2 depicts stall performance across different scenarios and configurations. Finally, Figure 6.3 shows F-Index of QoE between competing clients.

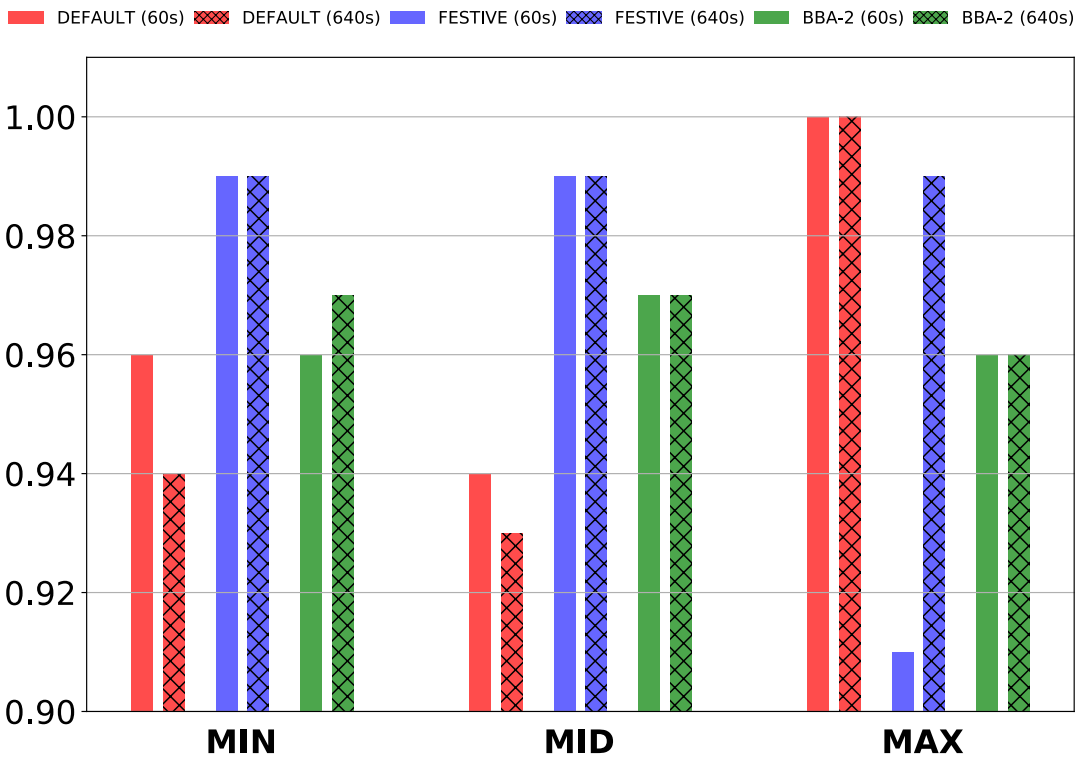


Figure 6.3: F-Index across different queue lengths

With the 1xBDP queue size (rule-of-thumb [VS94]) the link utilisation is 70% on average. The impact of removing "OFF" periods (640s application buffer) has a marginally positive effect on bandwidth utilisation (except for GPAC default algorithm, where utilisation improves by 9% for MIN case). Increasing queue length has a positive impact on utilisation across all algorithms. For MID queue sizes, the bandwidth hovers around 83-89%. In the extreme case of 30xBDP, clients achieve the highest bandwidth utilisation of around 89-96%.

All three algorithms request lower representation rates on average for the 1xBDP case. In the middle region (2-20xBDP), algorithms show improvement, requesting on average 900Kbps. When the queue is large enough to eliminate the packet-loss component at the bottleneck link, algorithms request higher representation

rates (averaging 1Mbps). The client buffer impacts the average representation rate significantly, resulting in higher rates when the client buffer duration is 60 seconds and vice versa. This holds for all the algorithms. These results have roots in the algorithm design. When using a large application buffer, clients access the network all the time, causing it to estimate lower throughput, while in the case of a smaller buffer, clients request segments at different times (randomised scheduling component of the FESTIVE) which leads to the estimation of larger throughput values. BBA-2 with a large buffer has slower up-switches because of the larger cushion regions, which forces a client to download more segments with lower quality before it can switch to higher representation rates.

Increasing the queue size has an insignificant impact on the instability of FESTIVE except for the extra large queue size (30xBDP). This implies FESTIVE would be more stable when the network does not drop packets. We also noticed that the client buffer duration affects FESTIVE's switching behaviour. For MIN queue size instability increases by 21% with larger application buffer. However, this difference diminishes as we increase the queue size. The queue size has a slight impact on the switching behaviour of BBA-2. On the contrary, a larger application buffer significantly improves BBA-2 switching behaviour. By its design, with larger buffer sizes, BBA-2 has a larger cushion region and hence performs fewer switches. When the application buffer is small, the cushion region for each nominal rate is also small, forcing the algorithm to switch too often. To illustrate, in the case of the 60 second buffer, clients are making on average 200 switches per clip. The clients request a new rate approximately every couple of segments, making them very unstable. Hence, it may be unfair to draw conclusions on BBA-2 performance with small or medium buffer sizes.

Overall user experience depends on the algorithm in use. For the GPAC default algorithm, QoE decreases with the larger queue, resulting in a massive drop to zero for the MAX queue size. The main reason is the stall performance as depicted in Table 6.2. Larger queue results in all clients experiencing stalls, with a very long stall duration (40 seconds on average). This negates the positive effect of a larger queue on the rate and bandwidth utilisation. This result is also intuitive as the latest studies indicate that users are most sensitive to stall performance when streaming a video. Similarly, FESTIVE shows a similar pattern. However, for MID queue size, QoE improves by 10% on average. However, MAX queue size resulted in a drop of stall performance for FESTIVE with smaller application buffer as a result of increased average rate and decreased instability (which causes the player to switch less). This behaviour leads to 6% drop for QoE compared

to MID queue size. Lower rate for the case with 640-second application buffer, improves stall performance (neither client experience a stall), resulting in 29% improvement of user experience compared to MIN queue size. When the queue length is big enough to accommodate all the outstanding packets in the network, queuing delay negatively impacts the stalls if the client algorithm is rate-based as in FESTIVE and GPAC default. In the case of BBA-2, QoE improves with queue size regardless of application buffer. Combines with a larger application buffer, BBA-2 achieves higher user experience compared to the small application buffer. This improvement is an outcome of improved switching behaviour due to increased cushion between subsequent rates. Also, a small buffer and queue length can lead to one client experiencing a relatively long stall.

Queue size has a marginal impact on F-Index as depicted in Figure 6.3. All algorithms achieve high values (over 0.9), indicating the fair share of resources. For the MAX queue size, GPAC produces a perfect score (1.0) as a consequence of all clients having a QoE value of zero. FESTIVE achieves an almost perfect fair share in all cases (0.99) bar combination of MAX queue size with small application buffer in which F-Index drops to 0.91. This result is a direct outcome of stall performance decline.

FESTIVE and BBA-2, represent two philosophies for rate picking approaches, rate-based and buffer-based. Although the scope of the analysis is not driven by any conclusion regarding what type of algorithm is better, we found that using a smaller buffer for BBA-2 leads to stability and fairness problems. This is due to shrinking the cushion region in the buffer. Hence, we expect that using a default 240 second [HJM+14] would be ideal for buffer-based strategies. That said, using a large buffer leads to inefficient use of resources if the user prematurely abandons the session.

In the rest of the analysis, the GPAC default algorithm is removed from the evaluation because most of HAS players estimate an average value for throughput samples rather than using raw throughput values for driving their decisions. In addition, the BBA-2 is configured with its default 240 second application buffer.

6.2.1 Looking beyond queue size

Dimensioning queue size depends on three components depending on the rule in use, namely RTT, link capacity and the number of clients competing for network resources. The previous section explored different queue sizes with these sub-

components fixed. In this section, however, we reverse the process, making the queue size constant and changing the components mentioned above. The reason for doing this is to generalise the results obtained in the previous section.

Firstly, using different bottleneck link capacities confirms the previous findings. For all link capacity values and algorithm type combinations, clients underutilise resources when the network queue length equals 1xBDP, using only on average 70% of the link capacity. For higher link capacity, average representation rate is not a good indicator of how well the clients utilise bandwidth resources. Adaptation algorithms switch cautiously to a higher rate, usually by one step (next subsequent rate). For higher bandwidths, a client needs more time to find the optimal rate. More unbiased metric is median value, as shown in Table 6.3. As the results show, both BBA-2 and FESTIVE stream with near optimal rate.

Table 6.3: Summary results for median representation rate (Kbps) across different bottleneck link capacity values

Bandwidth Link Capacity (Kbps)

(a) BBA-2

6	**12**	**18**	**24**	**30**
1040.0	1754.0	2976.0	3818.0	4267.0

(b) FESTIVE

6	**12**	**18**	**24**	**30**
771.0	1748.0	2387.0	3097.0	4267.0

The instability drops by 90% and 50%, as the capacity changes from 6Mbps to 30Mbps for FESTIVE and BBA-2, respectively. This happens because the client performs fewer switches as the gap between subsequent quality-rates increases, for the rates higher than 1Mbps.

Not surprisingly improvement in QoE metrics lead to overall QoE improvement. QoE score improves 4x and 3x as the capacity increases to 30Mbps for BBA-2 and FESTIVE, respectively. Furthermore, in all cases, clients share resources equitably achieving minimum 0.94 for F-Index.

Next, the impact of varying RTTs is investigated on the performance of FESTIVE and BBA-2. As previously mentioned, it is important to note that low RTT for video streaming is expected to be more common with the current trend of bringing the content distribution network closer to the edge of access networks.

Different RTT values are tested including $20ms$, $40ms$, $80ms$, and $160ms$. For

every RTT value, the experiment is repeated for the bottleneck queue size from 1xBDP to 10xBDP.

For large RTT ($80ms$ and $160ms$), BBA-2 and FESTIVE show insignificant changes in bandwidth utilisation, fairness, and instability in comparison to the $40ms$ RTT case. A slight drop in the average representation rate is observed as the RTT increases. Additionally, a noticeable drop in stalls is seen for $80ms$ case, and no stalls are encountered in the $160ms$ RTT case.

The $20ms$ RTT case shows a noticeable impact on both fairness and stall performance. More specifically, 30% FESTIVE clients encounter at least one stall for the 1xBDP case. As a result, the overall QoE score drops by 26% compared to $40ms$ case due to increased stall impairment. Similarly, BBA-2 clients show similar performance with 13% QoE drop compared to the default case. Furthermore, as some clients experience stalls, these re-buffering events negatively impact the overall fairness of the system. In particular, F-Index drops by 7% and 10% for BBA-2 and FESTIVE, respectively.

For larger queue sizes higher than 1xBDP, FESTIVE and BBA-2 clients show very close performance metrics similar to corresponding queue sizes for the $40ms$ case.

Results indicate that the streaming performance degradation could be avoided in the case with low RTT, by properly dimensioning the bottleneck queue size.

Finally, we run experiments with default settings, as shown in Table 6.1, only changing the number of clients from 2 to 8 for the network queue lengths equal to 1 and 2xBDP.

In all scenarios, using a larger queue size enables increasing network utilisation and achievable quality rate. For FESTIVE, the average quality rate increases by 19% on average, as the network queue size goes from 1xBDP to 2xBDP regardless of the number of clients sharing the link. Similarly, bandwidth utilisation increases by 17% and 12% for 1xBDP and 2xBDP, respectively, as the number of clients increases from 2 to 8. This confirms results from [CCPM13], where authors also show that FESTIVE utilisation increases with the number of clients. The instability metric doubles for both cases. These results are also consistent with results from [LZG+14] regarding instability trends (authors also have similar conclusion for unfairness metric, but they use Jain Fairness Index), where bandwidth overestimation and bitrate levels were pointed out as the possible root causes. Also by increasing the number of TCP connections, the requirement for a

1xBDP network queue lessens, causing higher bandwidth utilisation. As showed in [AKM04], a large number of TCP connections do not require a 1xBDP network queue.

For BBA-2 in the 1xBDP case, the utilisation increases by 8% as the number of clients increases from 2 to 8. BBA-2 achieves higher utilisation than FESTIVE for scenarios with a few clients (2 and 4). When 2xBDP is used for network queue size, increasing the number of clients has no effect on utilisation, resulting in same value across different cases. For both cases, instability increases linearly with the number of clients.

Intuitively increasing the number of clients sharing resources has a negative effect on the QoE score. Table 6.4 shows QoE values for different number of clients and queue lengths.

Table 6.4: QoE values across different number of clients
(V_i **-** i **HAS clients sharing bottleneck link**)

(a) FESTIVE

	1xBDP				**2xBDP**			
#	V_2	V_4	V_6	V_8	V_2	V_4	V_6	V_8
QoE	1.56	0.98	0.81	0.70	1.87	1.02	0.89	0.74

(b) BBA-2

	1xBDP				**2xBDP**			
#	V_2	V_4	V_6	V_8	V_2	V_4	V_6	V_8
QoE	1.70	0.79	0.64	0.48	2.00	1.07	0.71	0.60

However, moving from 1xBDP to 2xBDP increases overall QoE for all scenarios regardless of chosen adaptation algorithm. Improvement in QoE varies between 5% - 35%. For FESTIVE, the conservative approach in picking higher rate (streaming k segments encoded in k bitrate before moving to $k+1$ quality) limits the impact of 2xBDP compared to more aggressive rate selection in a case of BBA-2. Second, switching impairment (in a case of Clay QoE represents a standard deviation of average rate) increases for larger queue additionally limiting overall QoE improvement. Third, the spacing between subsequent rates plays an important role in the overall QoE score. For example, in the case of four clients competing for 6Mbps bandwidth, clients stream at 1Mbps rate quality. However larger queue enables clients to stream at 1.7Mbps improving overall QoE significantly (by 35%). For other cases, a gap between subsequent rates is smaller (around 30%) resulting in less noticeable improvement.

6.3 Experiments with mixed traffic

The bottleneck link may be shared among different types of traffic, including video, and interactive, delay-sensitive applications. Hence, next set of experiments considers scenarios when multiple heterogeneous applications share a bottleneck. In particular, a mixture of HAS and web clients are considered, as they represent the most popular types of applications in today's Internet.

All experiments consider six clients. In the rest of the analysis, we use M_n to refer to sharing scenario with n web clients and $6 - n$ video clients. The default simulation parameters are used as shown in Table 6.1, for FIFO queue. In addition to FIFO, AQM techniques are analysed as well, namely CoDel and FQ-CoDel.

Comparing QoE metrics (bandwidth utilisation, average representation rate, average instability and stall performance), there is no significant difference between different queuing disciplines when clients stream with either BBA-2 or FESTIVE adaptation algorithm. the biggest difference occurs for FESTIVE algorithm coupled with AQM, with average representation rate lower by 6% compared to FIFO.

Finally, the impact of AQM has a negligible effect on the overall QoE score, as depicted in Figure 6.4. For FESTIVE, AQM lowers QoE, while with BBA-2 trend is the opposite, with an average difference around 1% compared to FIFO.

In addition to QoE score, Table 6.5 shows level of fairness among competing clients. Similar to QoE, there is no significant difference in how QoE is distributed among clients for different queuing disciplines.

Table 6.5: F-Index across different queuing disciplines and adaptation algorithms

Algorithm \ Queuing Disc.	**FIFO**	**CoDel**	**FQ-CoDel**
BBA-2	0.96	0.95	0.96
FESTIVE	0.99	0.99	0.99

6.3.1 Impact of web traffic on video performance

Before analysing the impact of video traffic on QoE performance of web clients, video performance is compared with and without web sharing clients. As a notation, V_k refers to k video clients sharing a bottleneck link. For example, when comparing mixed and video cases, M_n is analogous to V_{6-n}.

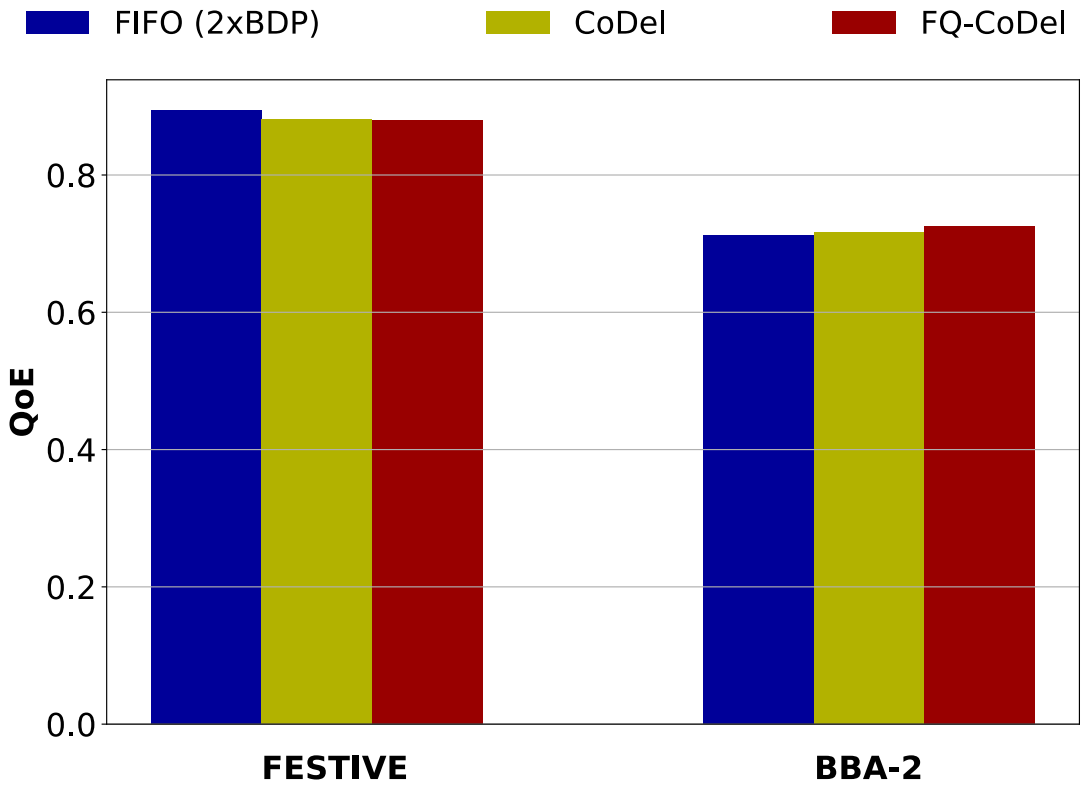


Figure 6.4: QoE performance for different queuing disciplines and adaptation algorithms

Figure 6.5 illustrates performance of QoE metrics for different combinations of video and web clients across 1xBDP and 2xBDP cases and adaptation algorithms.

Intuitively, the average representation rate increases with the number of web clients as fewer HAS clients are competing for the link. However, the perceived average quality rate noticeably decreases as the number of web clients increases in comparison to corresponding V_{6-n} scenarios.

Instability is scenario-dependent as web clients may change this metric positively or negatively in comparison to V_{6-n} scenarios. Such scenario dependency is due to the interaction between the link capacity, the number of clients, and available video encoding rate. To illustrate, instability changes differently across different M_n scenarios in comparison to V_{6-n} scenarios. In the M_3 and V_3 scenarios with 1xBDP for FESTIVE, median quality rate is 1Mbps. In M_3 case, 700Kbps generated by web clients (237Kbps per client), fills the gap between 1065 and 1777Kbps. This forces all the video clients to stream at the 1Mbps. Without the web traffic, case V_3, two clients select higher rates, 1777 and 2335Kbps, forcing the third client to use 1Mbps. This increases instability, which seems counter-

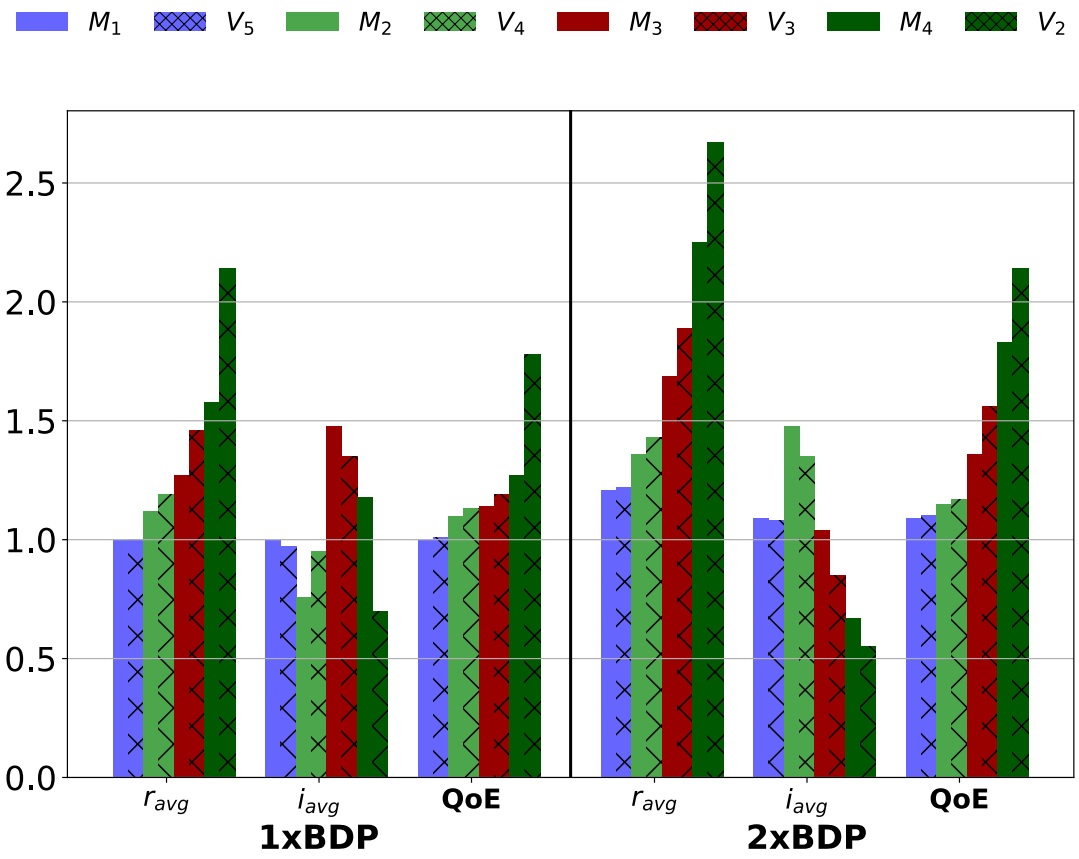
intuitive. However, FESTIVE's probe component forces clients to periodically select a higher rate as a function of rate. For lower rates, the algorithm will try next higher rate more often. This explains why instability increases by 10%. For the 2xBDP case, the same pattern can be observed, but because the larger network queue increases available throughput, the aforementioned interpretations shift to scenarios M_2 and V_4, respectively. BBA-2 relies on buffer levels to make a decision. When video clients operate in a region around 1Mbps, the amount of web traffic helps to decrease instability. This is a consequence of slower buffer filling, which essentially causes clients to be more stable.

As a result, overall QoE decreases with the introduction of web clients. However, for a smaller number of web clients (less than three), this difference is negligible. Largest difference occurs when two video clients share resources with four web clients with QoE dropping by 29% and 15% for 1xBDP and 2xBDP, respectively in case of FESTIVE. Similarly, BBA-2 shows the same trend with a 27% and 20% drop in QoE for the same cases.

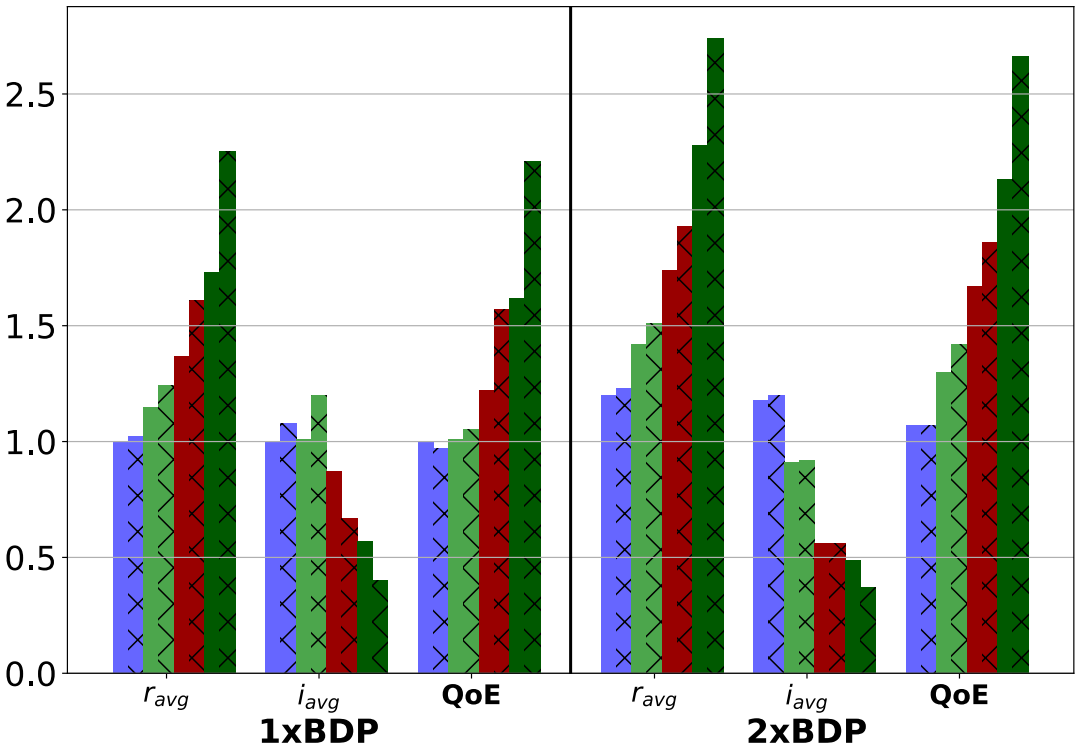
Next, different active queue management scheduling techniques (CoDel and FQ-CoDel) are compared for mixed-case traffic scenarios, where two types of clients share the bottleneck link.

For the FESTIVE adaptation algorithm, the average representation rate is higher in the case of FIFO than CoDel and FQ-CoDel. However, the difference is slim. When clients are streaming with BBA-2, average throughput is similar across different queuing techniques, resulting in negligible difference. However, for the M_4 case, AQM techniques achieve higher average representation rate (5%). This increase is countered in instability, with clients switching more. QoE performance is scenario dependent. In most cases, the impact of AQM has a negligible effect on QoE, with FIFO producing higher QoE in a majority of cases, as depicted in Figure 6.6a. Still, for M_4 case, AQM achieves 5% higher QoE compared to FIFO when clients use BBA-2.

Similar, Figure 6.6b shows fairness among competing users. All queuing disciplines achieve consistent fairness across the majority of cases when streaming with a FESTIVE algorithm. BBA-2 exhibits similar performance with slightly lower fairness.

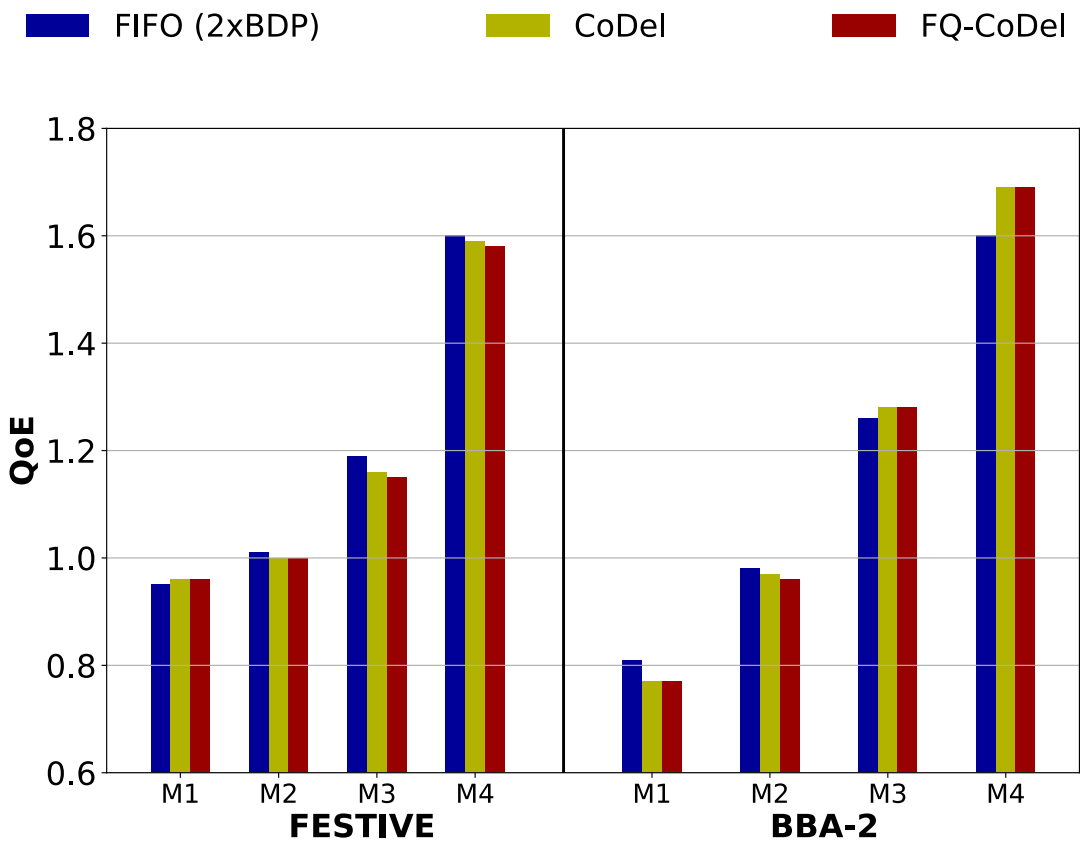


(a) FESTIVE

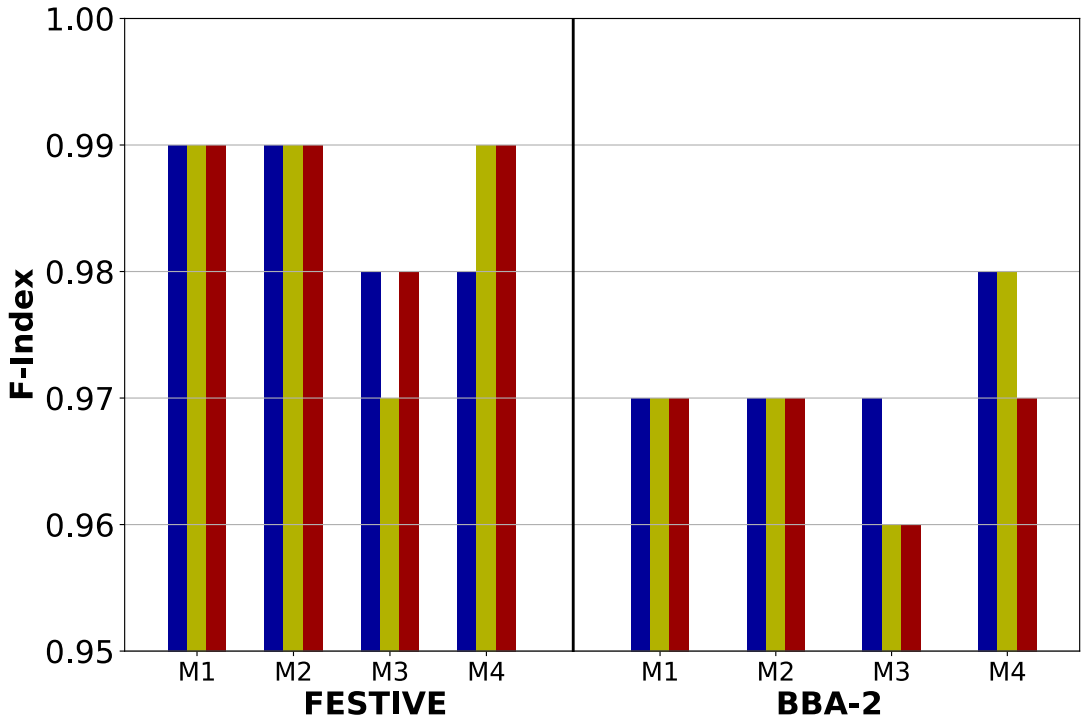


(b) BBA-2

Figure 6.5: Video performance in presence of web clients for FIFO queuing discipline (all values are normalized to M_1 scenario values for different QoE metrics)



(a) QoE



(b) F-Index

Figure 6.6: Video QoE and fairness in mixed traffic scenarios for different queuing disciplines and adaptation algorithms

6.3.2 Impact of video traffic on web performance

First, M_6 scenario is considered for different queue lengths; i.e. six web clients sharing the link. Increasing the queue size, reduces the PLT as the network queue increases from 1x to 2xBDP. As a result QoE increases by 2%, as depicted in Figure 6.7. Small network queues drop packets more frequently, causing TCP to retransmit lost packets. This adds to the overall delay, causing the user to abandon more websites. Note that our user abandons the web page if it doesn't load in 15 sec [LWD10]. Also, with less retransmitted packets allows TCP to achieve higher throughput. This result indicates that larger queues could also benefit web clients sharing a bottleneck link. For remaining scenarios M_1 - M_5, web QoE is scenario dependent. For example, the M_2 scenario with 1xBDP, PLT values drop for FESTIVE and BBA-2 increasing QoE as well. Observing the video client's rates, four video clients have similar rates as for M_1 scenario, which has five video clients beside one web client. Because of the gap between 1 and 1.7Mbps rate, four clients don't have enough resources to compete for the next higher rate. As a result, the network queue is less occupied with video traffic, allowing web clients to achieve better user experience.

Finally, he impact of video traffic on web QoE for modern AQM techniques is evaluated. Figure 6.7 shows web QoE performance across different competing scenarios and adaptation algorithms. For M_6 scenario, both AQM techniques result in a 10% decrease for PLT. This translates to 4% increase for web QoE. In the case of having five video clients and one web client sharing a bottleneck link, QoE drops by 6.5% and 10.5% on average, for BBA-2 and FESTIVE (e.g. PLT increases by 33% and 10% on average), respectively across different AQM techniques. The impact on web QoE depends on a choice of adaptation algorithm. As a more pugnacious algorithm, BBA-2 has a more significant negative impact on QoE than FESTIVE. For the majority of scenarios, FQ-CoDel achieves higher QoE than CoDel. For BBA-2 adaptation algorithm, the variation of QoE value is scenario dependent. The gap between subsequent rates results in a lower value for QoE. Looking carefully into scenarios M_1, M_2, and M_3 the following pattern is observed: 75th percentile of representation rate is equal to 1.7Mbps for three out of five video users (with rest of clients having 1Mbps 75th percentile). Similar, for the M_2 case all clients have 75th percentile equal to 1.7Mbps. Furthermore, two of three clients have 75th percentile equal to 1.7Mbps in M_3 case (remaining one having 2.3Mbps). This observation draws a natural conclusion: Decreasing the number of video clients, their streaming rate stays the same resulting in a less

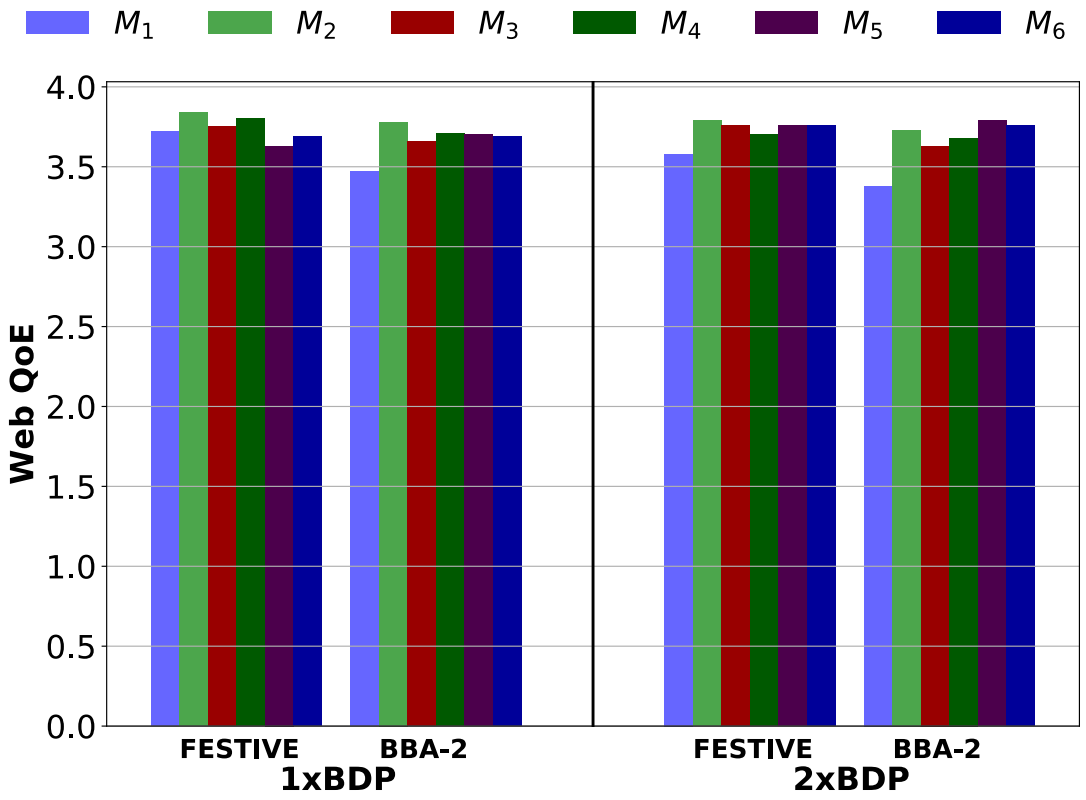


Figure 6.7: Web QoE performance in multi-traffic scenario for FIFO queuing discipline

occupied queue by video traffic. The gap between 1Mbps and 1.7Mbps is uniquely higher (70%) than between rest of rates (∼30%). Larger gap limits video clients in streaming with the higher rate even as the number of clients decreases.

Similar, for the M_4, both clients are trying to stream with 3Mbps representation rate. As a result, web client delay increases. However, in M_5 case QoE increases, although video client streams with the highest rate. In this case, 4Mbps is the highest possible representation rate, allowing more resources for web clients.

For the FESTIVE adaptation algorithm, the difference in QoE values for different scenario cases is notably less evident compared to BBA-2. However, both algorithms have a similar trend across different cases for both CoDel and FQ-CoDel.

6.4 Experiments with heterogeneous RTTs

In practice, clients may not have homogeneous RTT values. For example, with Content Delivery Networks (CDN) nodes close to the access network, there will be

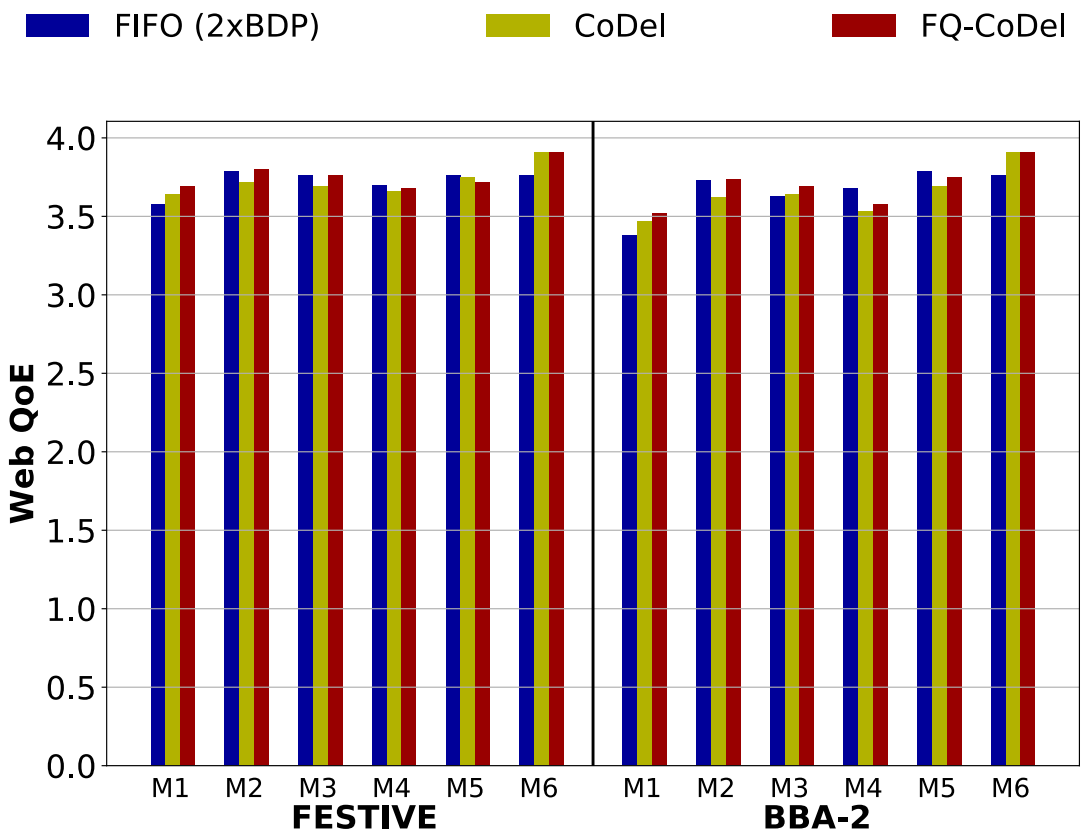
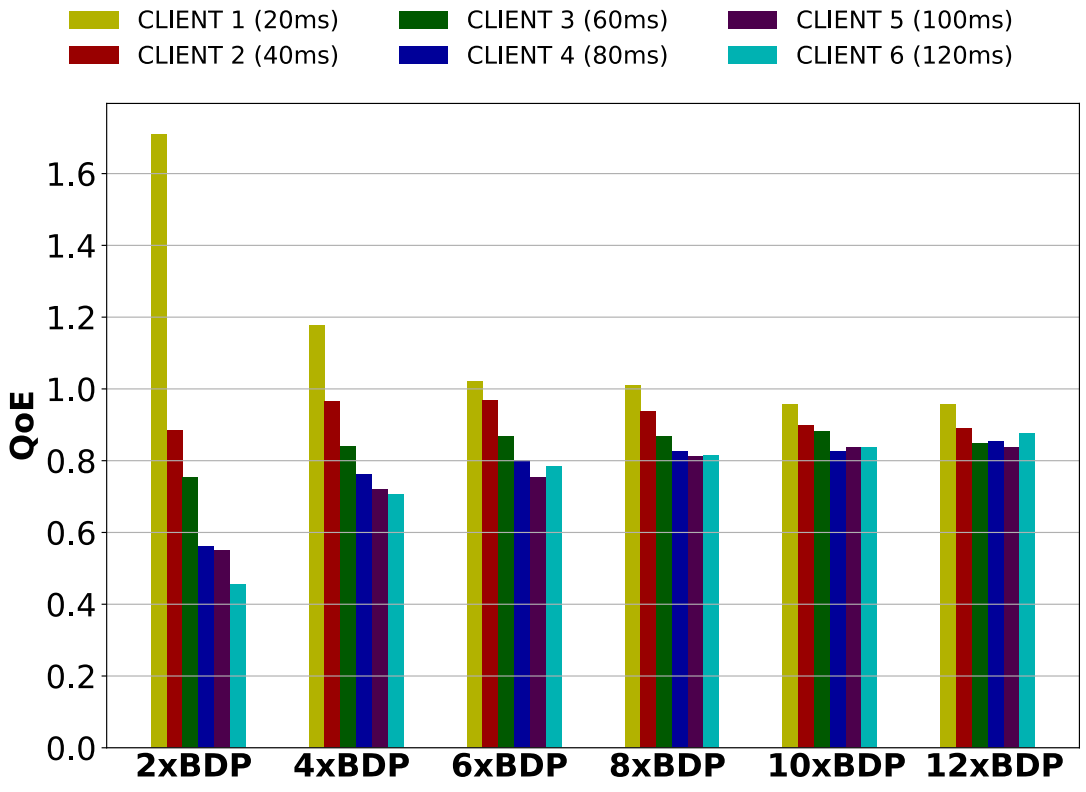


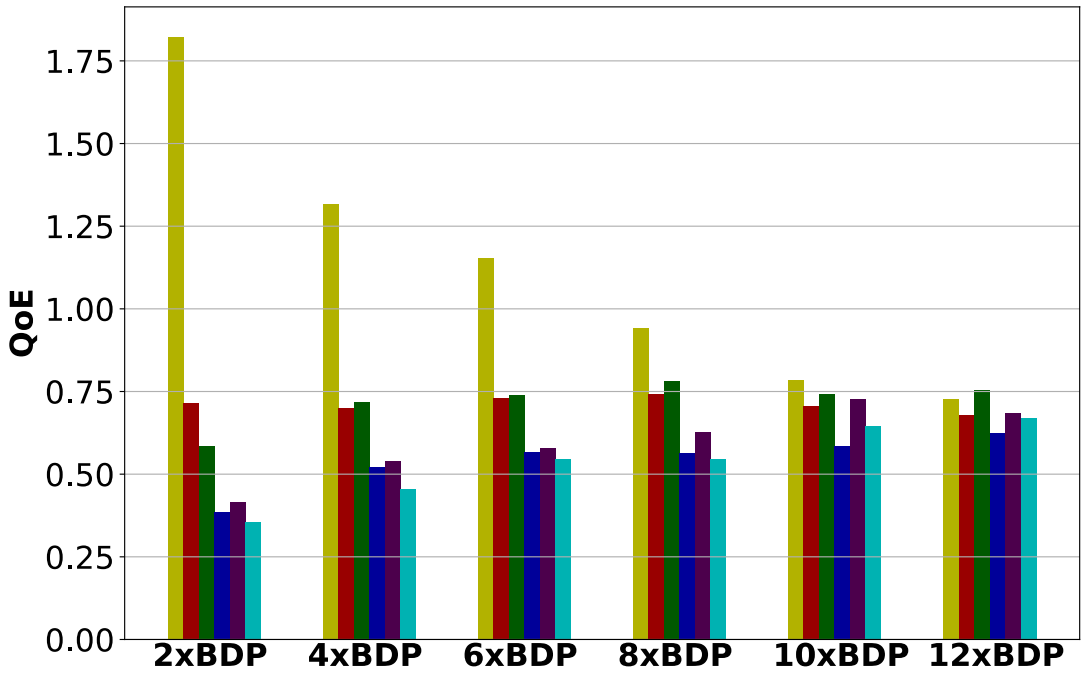
Figure 6.8: Web QoE performance in multi-traffic scenario for AQM queuing discipline

lower RTT values for customers of the corresponding video service when compared to other services that are using further away servers. Since the TCP performance depends on perceived RTT values, we conduct experiments with clients having different RTTs while competing over the same bottleneck link. Thus we tested BBA-2 and FESTIVE with 240 and 60-second client buffers respectively, and for six clients, competing for resources, with their RTT values set to 20, 40, 60, 80, 100, and 120ms. Similarly to previous sections, we repeat experiments with different network queue lengths. Also, we test preceding AQM techniques against heterogeneous RTT values. All network queue lengths are estimated based on the RTT of the first client (20ms). Hence, 4xBDP for the first client is a 2xBDP queue for the second client, which has 40ms RTT.

Figure 6.10 shows negative impact of heterogeneous RTT on QoE performance for competing clients. For 2xBDP case, both algorithms achieve the lowest fairness among multiple clients. Fairness drops to 0.83 and 0.79 for FESTIVE and BBA-2, respectively. Low fairness comes from clients with low RTT having better QoE than clients with higher RTT, as depicted in Figure 6.9a and 6.9b. For the



(a) FESTIVE



(b) BBA-2

Figure 6.9: Queue Length impact on QoE in heterogeneous RTT environment

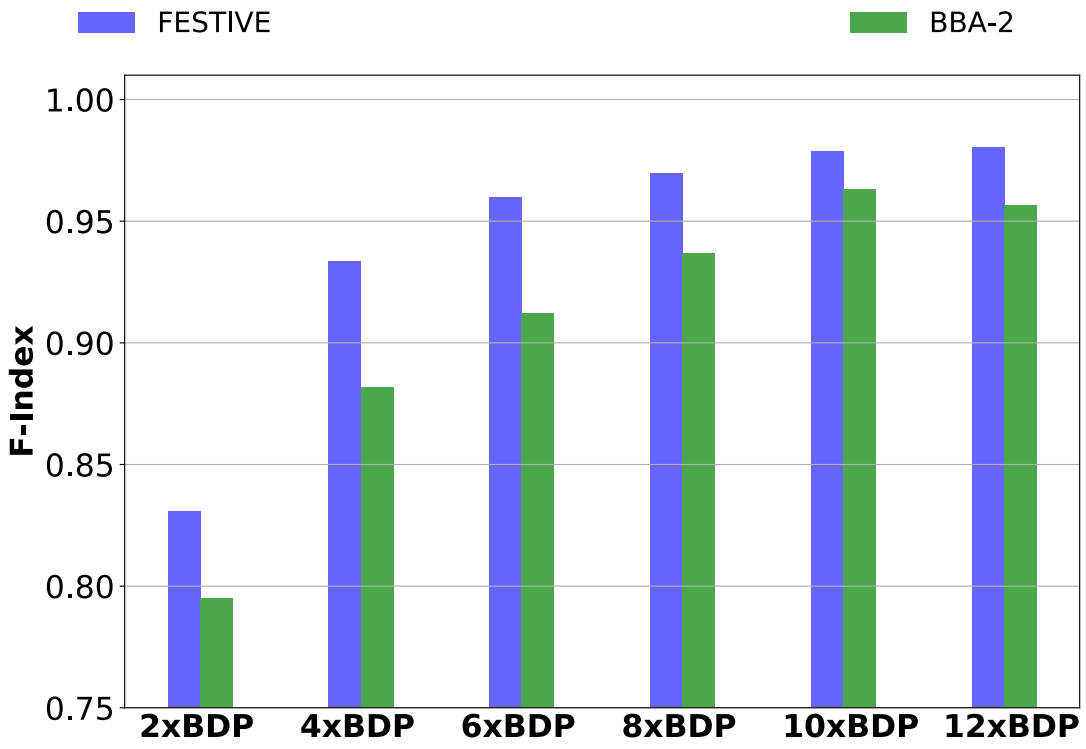


Figure 6.10: Queue Length impact on Fairness in heterogeneous RTT environment

2xBDP case, a client with 20ms RTT achieves 5x and 4x higher QoE compared to a client with 120ms, for FESTIVE and BBA-2, respectively. However, increasing the queue size counter effect of heterogeneous RTTs, increasing overall fairness. For queue lengths larger than 8xBDP, both algorithms achieve high fairness (>0.95). All clients experience similar QoE score (0.7-0.8) for both algorithms.

Table 6.6: Average instability across different network queue lengths (products of BDP) for heterogeneous RTTs

Algorithm \ Queue Length	**2x**	**4x**	**6x**	**8x**	**10x**	**12x**
BBA-2	0.068	0.065	0.06	0.058	0.058	0.059
FESTIVE	0.157	0.11	0.11	0.10	0.10	0.10

Table 6.6 shows average instability for both BBA-2 and FESTIVE algorithms. Queue length has no significant impact on instability when the adaptation algorithm is BBA-2. For FESTIVE, except for 2xBDP queue length, the average instability does not change across different queue lengths. For the 2xBDP case, client 6 has a larger number of switches, which drives overall average instability. An increased number of switches is a consequence of FESTIVE stability function. The average rate for client 6 is around 500Kbps. For lower rates, FESTIVE is

less cautious when switching-up. As the average rate increases, the number of switches goes down, and all the clients have a similar number of switches.

As results show, clients with the smallest RTTs have preferential access to network resources dominating all other clients with higher RTTs. This result is intuitive, as all the clients use TCP as a transport protocol. Long-term throughput (in steady state) of TCP connection can be expressed as [MSMO97]:

$$Rate_{TCP} = \frac{1}{RTT}\frac{1.22 \times MSS}{\sqrt{p}} \tag{6.1}$$

where MSS is maximum segment size, and p represents probability of random packet loss. Equation (6.1) shows that rate is inversely proportional to RTT which confirms our results. By increasing queue length, queuing delay increases, adding to overall RTT. Queuing delay helps to level the field for the client with higher RTT.

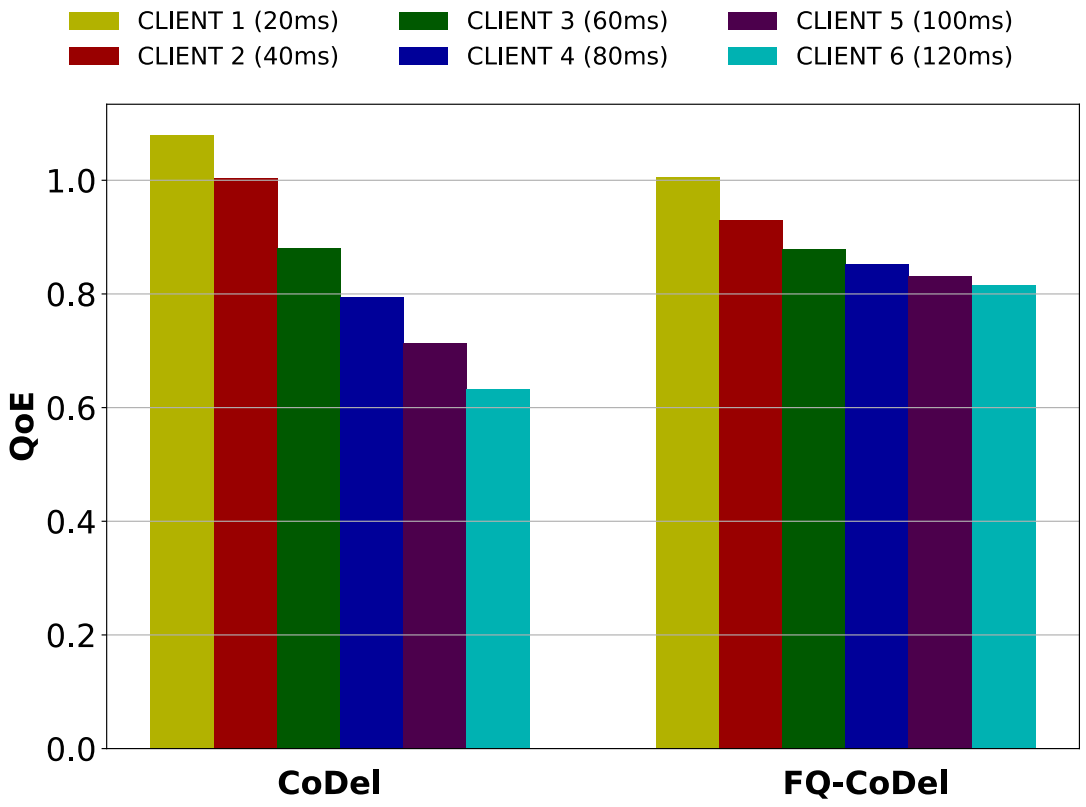
Next, we analyse the QoE performance in an environment with heterogeneous RTTs coupled with AQM. Figure 6.11 depicts the QoE of competing clients for different adaptation algorithms and two queuing disciplines, CoDel and FQ-CoDel. Flow isolation helps in reducing the unfair share of resources, as the difference between clients with highest and lowest RTT (120 and 20ms) is 50% 23% for BBA-2 and FESTIVE, respectively. However, in the case of CoDel without flow isolation, this difference increases by 170% and 46% for BBA-2 and FESTIVE, respectively.

Overall, FQ-CoDel achieves high system fairness, as depicted in Table 6.7. CoDel archives high fairness when coupled with FESTIVE adaptation algorithm, while more aggressive adaptation strategy results in lower overall fairness.

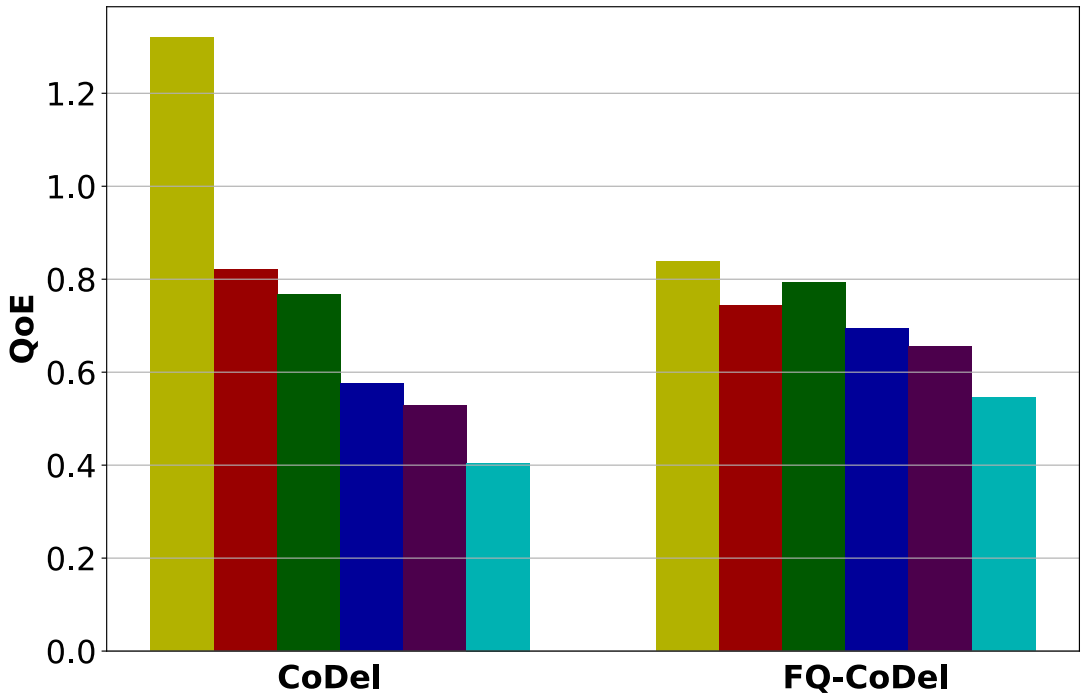
Table 6.7: F-Index across different AQMs for heterogeneous RTTs

Algorithm \ AQM	**CoDel**	**FQ-CoDel**
BBA-2	0.88	0.95
FESTIVE	0.94	0.97

Flow isolation helps in reducing instability when compared with using only CoDel, as depicted in Table 6.8. However, compared to the homogeneous RTT case, there is no significant difference when using BBA-2 as adaptation algorithm. For the FESTIVE case, instability increases by 10% for both CoDel and FQ-CoDel.



(a) QoE score for each client and FESTIVE adaptation algorithm



(b) QoE score for each client and BBA-2 adaptation algorithm

Figure 6.11: QoE performance in heterogeneous RTT environment with AQM mechanisms

Table 6.8: Average instability across different AQM mechanisms for heterogeneous RTTs

Algorithm \ AQM	**CoDel**	**FQ-CoDel**
BBA-2	0.06	0.05
FESTIVE	0.11	0.1

6.5 Discussion

This section provides a summary and discussion of key results, seeking to offer guidelines for network practitioners charged with configuring network routers to manage a mix of HAS and non-HAS traffic.

For the first hypothesis "The well-known 1xBDP rule-of-thumb for dimensioning network queues can achieve full link utilisation for multiple HAS video clients" experiments indicate that the rule-of-thumb queue size leads to relatively lower bandwidth utilisation and lower average streaming rates and has no performance merit for other metrics. As result the first hypothesis is rejected.

For the second hypothesis "Increasing router queue length will improve fairness among competing HAS video clients with heterogeneous RTTs" performed experiments show that employing larger network queue sizes assists the system in improving its fairness for video clients with heterogeneous RTT delays. Note that this is a critical issue as some content providers site their CDN distribution nodes close to, or within, the network of the Internet service provider; As result the second hypothesis is accepted.

Finally, for the third hypothesis, "Use of AQM techniques will protect web client performance (in particular page-loading time) when sharing resources with HAS video clients", experiments show that using AQM techniques improve page-loading time when web-only clients share a bottleneck link, while in a mixed traffic scenario, the positive effect of CoDel and FQ-CoDel on page-loading time is negligible and is more dependent on the type of adaptation algorithm and number of video users. As a result, the third hypotesis is rejected.

Empirical study performed in this chapter yields numerous additional interesting observations when multiple video clients compete for the bottleneck resource.

Following conclusions are made:

- The performance degradation of low RTT can be avoided by using a queue

size larger than 1xBDP.

- Using AQM instead of FIFO does not significantly impair video performance.
- In the case of heterogeneous RTT delays, CoDel shows the same weakness as FIFO, while flow isolation (FQ-CoDel) improves fairness.

When multiple web and video users share a bottleneck link, the following observations are made:

- Increasing the bottleneck queue size helps web clients that compete with HAS clients by increasing their throughput and decreasing the number of excessively delayed web pages, but would have a scenario-dependent impact on the web page load time.

Based on these results, the rule-of-thumb 1xBDP queue size does not seem to be a good choice for HAS clients sharing a bottleneck, but rather a medium queue size strikes a sweet operating point for video clients with homogeneous RTT times.

Sharing network resources between the web and HAS clients can be a detriment to web QoE. Deploying AQM techniques for alleviating these effects seems ineffectual, as the page loading time mostly depends on the gap between adjacent representation rates and type of adaptation algorithm.

The effects of queue size on live HTTP adaptive streaming latency performance is a compelling research problem as end-to-end latency is an important metric when evaluating user experience. End-to-end-latency is defined as the difference between the time point when the live event takes place and when it is played back to the user. The leading factor contributing to end-to-end latency is chunk duration. Minimum latency is bounded to the delivery time of one chunk plus encoding and network delay. However, chunk duration is usually a couple of seconds long resulting in potential end-to-end latency of order of tens seconds [WS14]. Intuitively, shorter chunks would improve latency values at the cost of signalling overhead (i.e. each chunk is preceded with a request from the client) and coding efficiency. On the upside, HTTP/2 push-based capability alleviates this issue allowing sub-second chunks [vdHPW+18]. Alternatively, HTTP chunked transfer encoding, a technique that allows the chunk to be generated and delivered simultaneously in small pieces (called chunks), can be used with existing HTTP 1.1 [BTBZ19]. In our experiments, the average queuing delay for large queue sizes is up to 200 and 226ms for BBA-2 and FESTIVE, respectively. For live

streaming using chunks, this delay would need to be allowed for in calculating the end-to-end delay budget.

6.6 Conclusion

This chapter explores performance interactions between multiple HAS and non-HAS clients sharing network resources from the perspective of network queue design in a realistic environment. These experiments are based on real traffic and realistic behaviour models for video and web clients.

The systematic empirical analysis yields several important practical results. The well-known "rule-of-thumb" (1xBDP) for network queue dimensioning causes underutilisation (70%) when multiple HAS clients share a bottleneck. Larger queues, e.g. 2xBDP, can improve utilisation and quality by 15% on average. Also, for median queue size, overall QoE improves by 12% on average. Larger queues also help in improving system fairness for clients with heterogeneous RTTs. However, larger queues can negatively impact delay-sensitive traffic, causing *bufferbloat* phenomenon. While modern AQM techniques promise low delay and high utilisation and are application-agnostic, our results show that their performance is mostly scenario-dependant and would vary depending on bitrate distribution, video adaptation algorithm and offered web traffic load. Replacing FIFO with AQM does not improve PLT significantly, in which AQM is unable to counter the bandwidth-hungry nature of HAS flows. This striking result opens a space for developing a new AQM mechanism that can adequately protect delay-sensitive flows in cases when they share network resources with video streams.

Chapter 7

A solution for experience-oriented adaptive queuing

The previous chapter analysed the performance of multiple HTTP adaptive streaming (HAS) clients sharing network resources. Furthermore, exhaustive experiments were performed to quantify interactions between HAS and Web clients sharing a bottleneck link. The following key observations were made:

- Larger queues (i.e. adding queuing delay) improve HAS performance (Quality of Experience (QoE))
- Active Queue Management (AQM) disciplines (i.e. Controlled Delay Management (CoDel) and Flow Queue (FQ)-CoDel) are ineffective in achieving low page-loading time when web clients compete with HAS clients for the resources

In the case of heterogeneous traffic with distinct requirements, isolating traffic types and design application-aware scheduling are recommended. In [DSBTB16], the authors propose a two queue solution for correspondent coupling between traditional Transmission Control Protocol (TCP) and Data Centre TCP. For HAS traffic, results from the previous chapter suggest that extended video packet queuing would be preferable to packet dropping. This recommendation can be integrated into various scheduling disciplines, such as the Deficit Round Robin or Earliest Deadline First to achieve the best user experience for different applications. Typically, the number of traffic classes should remain small (2-3) for practical implementation.

This chapter presents ENDUE, an experience oriented adaptive two queue discipline for heterogeneous traffic based on the observations made above to improve the user experience for various traffic.

A key result is that isolating web and video traffic in conjunction with a delay-based adaptive scheduler improves page-loading time by up to 80% compared to state-of-the art CoDel AQM discipline with the insignificant impact on video performance.

The Chapter starts with the rationale for the ENDUE design elements through experimental analysis, followed by the comparison between ENDUE, First In First Out (FIFO) and state-of-the-art AQM CoDel queuing disciplines. The experimental methodology used for experiments is first presented.

7.1 Methodology

Similar to the previous chapter, we examine multiple HAS and web clients sharing a bottleneck link. We compare the proposed ENDUE discipline with the CoDel AQM discipline across various combinations of HAS and web clients.

7.1.1 Experimental testbed

The same testbed framework as described in Section 3.2.2 is used. There are N clients sharing a bottleneck link and requesting their data from a remote HyperText Transfer Protocol (HTTP) server. Video and web content is stored on HTTP servers. The role of the bottleneck link and network elements is to emulate scenarios with different queue scheduling techniques. Performance metrics for different applications are collected at the client side. Table 7.1 shows default values and ranges for key parameters used in the experiments.

Table 7.1: Different Parameter values for experiments

Parameter Name	**Default Value**
N (number of clients)	6
RTT (ms)	40
C (Mbps)	6
$Client\ Buffer$(s)	15,120
$Adaptation\ Algorithms$	Elastic, Logistic
$Network\ Queue$	FIFO, CoDel, ENDUE

In all experiments, there are six clients, representing a typical home environment. The number of web clients is n, where $n \in 1..5$, and $6-n$ HAS clients. In the rest of the text, as in the previous chapter, M_n refers to a sharing scenario with n web clients. All clients start streaming/browsing at the same time. The main reason is dashc player which crashes unexpectedly if multiple players start at random times.

7.1.2 HAS video client

For the video player, dashc [RZS18b] is selected instead of GPAC. Dashc emulates a HAS video player by adopting all the player's logic, except the option to decode the video content. Dashc is more flexible tool and easier to extend with new adaptation algorithms and support running larger number of clients. The following adaptation algorithms are evaluated: ELASTIC and LOGISTIC, as explained in Section 3.2.3.2. These algorithms are comparable to previously used algorithms in Chapter 6 but are newer released algorithms [CCPM13].

For video traffic, the same approach is used as in Chapter 6. More details about video dataset can be found in Section 3.2.3.1. The segment duration for all clips is 4 seconds. All the experiments last for the 5 minutes which allows using both video QoE models (Section 3.2.3.4).

7.1.3 Web client

To accompany dashc emulation HAS player, we write a custom web browser emulation tool, SPEED. Section 3.2.3.3 detailedly explains the SPEED tool. In a nutshell, SPEED uses *aiohttp* library for sending parallel TCP connections similar to the real web browser. In the first step, SPEED downloads the main object, after it downloads the remaining inline objects using multiple connections. Unlike a real web browser, SPEED does not decode main and inline objects, making it unusable for downloading real web pages. To overcome this issue, *dummy* web pages are created and stored on the web server. We create sizes of web page content based on an experimental analysis of the million web pages from the Internet [PMT12].

7.1.4 Key performance metrics

QoE metrics for video and web clients are summarised in Table 7.2. Stall performance is omitted as in most cases, all algorithms achieved a stall-free session. However, when appropriate stall performance is included.

The results shown represent the average of ten runs. In every run, each client starts at the same time. For web clients, the user randomly selects a web page and reads a page for some time (as explained in the previous section).

Table 7.2: QoE Metrics Notation for Video and Web clients

Video QoE Metrics

Metric	Summary
Bitrate	Average bitrate
Stability	Average stability, equal to $1 - i$ with i being instability as defined in [JSZ14]
QoE	Geometric mean of Clay and Yao QoE, Section 3.2.3.4

Web QoE Metrics

Metric	Summary
PLT	Page loading time for web users, Section 3.2.3.4
QoE	QoE for web user experience as a function of PLT, Section 3.2.3.4

7.2 ENDUE's design

Results in Chapter 6 indicate that the first step in designing a queuing discipline is to separate video and web flows. The reason is straightforward; web flows are delay-sensitive traffic, while video flows, as indicated by results in the previous chapter, are delay-tolerant traffic. Figure 7.1 depicts the simplified architecture of the ENDUE discipline. The queuing discipline comprises of two components, queue for storing and managing packets of different flows, and scheduler for managing packets based on some policy. The goal of queue disciplines is to limit queue length (and thus queuing delay) for web traffic and allow larger delays for video traffic. In addition, the adaptive scheduler supports this goal by increasing available bandwidth for web traffic and increasing video queuing delay at the same time.

With the ENDUE, there are two main requirements for queue discipline selection. First, queue discipline should be simple, and second, it should support easy tuning to change queue behaviour according to traffic characteristics. Similar, scheduler should be efficient, easy to implement and fair (in terms of allocating bandwidth among different traffic classes).

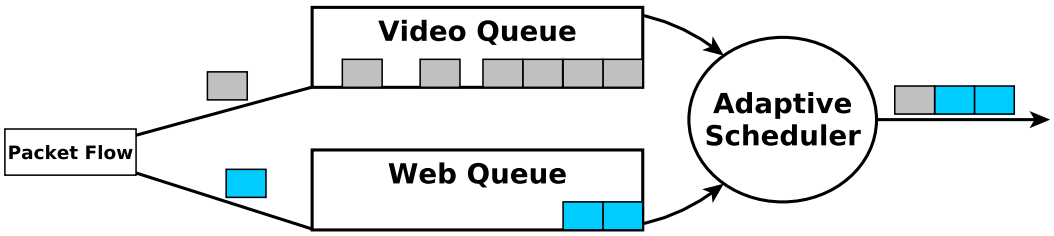


Figure 7.1: ENDUE's simplified architecture

For scheduling packets, Deficit Round Robin (DRR) [SV96] is used. DRR belongs to a class of Latency Rate (LR) servers [SV98]. The behaviour of LR class servers is determined by latency and allocated rate. DRR is a simple scheduling discipline that achieves an almost perfect fairness among different classes, and it is efficient (requiring $O(1)$ to process a packet). DRR implementation is based on the usage of *quantums*. In each round one *quantum* of bytes are added to every class. It allows each class to send the allocated number of bytes during the round. If the packet size is smaller than the allocated number of bytes, it adds the remaining bytes to the next round. If the packet size is larger than the allocated number of bytes, then the packet needs to wait for the next round and allocation of a new *quantum* of bytes to the remaining bytes. To achieve $O(1)$ operation per packet, the *quantum* size needs to be larger than the Maximum Transmission Unit (MTU) [SV96]. For *quantum*s smaller than the MTU, a modified DRR is proposed to preserve $O(1)$ complexity [LMS04].

7.2.1 Selecting the queuing discipline

This section starts by analysing different queuing disciplines for the video and web traffic. In analysis, two queuing disciplines are considered, FIFO and CoDel. Both disciplines are relatively simple and easy to tune. The function of these queues is to limit web queuing delay and keep it lower than video queuing delay. For simplicity, equal *quantum*s are assigned for both queues (scheduler policy is fixed). This results in equal bandwidth share across different M_i scenarios.

Figure 7.2 shows a boxplot for Page Loading Time (PLT) of web clients for

LOGISTIC and ELASTIC adaptation algorithms. Similarly, Table 7.3 shows average video QoE for the same setup and algorithms.

Both disciplines achieve the same trend across different M_i scenarios. PLT increases with the number of web clients. In the case with ELASTIC algorithm, median PLT increases from $1943ms$ to $2853ms$ for $M_1 - M_5$ scenarios. Similar, for LOGISTIC, median PLT increases from $1887ms$ to $3244ms$. While this seems counter-intuitive at first, traffic isolation is the main culprit of this result. With a fixed bandwidth allocation per traffic class, web users get only half of bandwidth on average. As the number of web users increases, more users are competing for the bandwidth increasing overall PLT. In the single queue approach, web clients have access to full bandwidth capacity. Similar, video QoE improves as the number of video users decrease, allowing video clients to stream at the higher rate.

Table 7.3: Average video QoE score across various M_i scenarios with LOGISTIC and ELASTIC adaptation algorithms

LOGISTIC

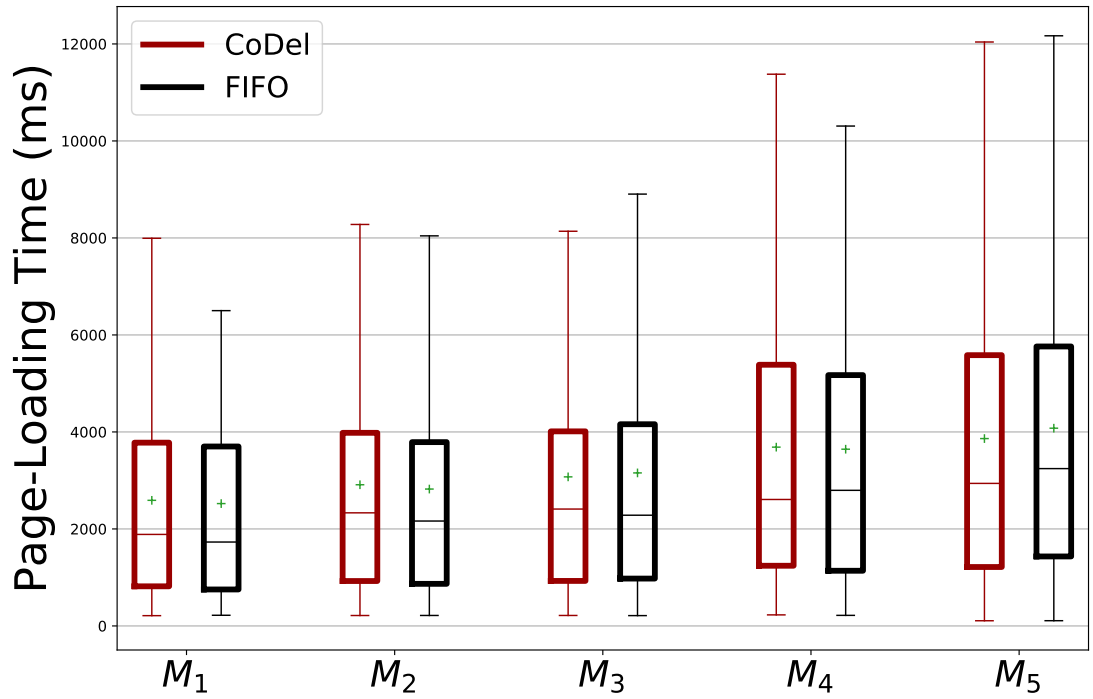
	M_1	M_2	M_3	M_4	M_5
CoDel	8.62	8.59	8.95	10.0	11.51
FIFO	8.5	8.52	9.0	10.1	11.74

ELASTIC

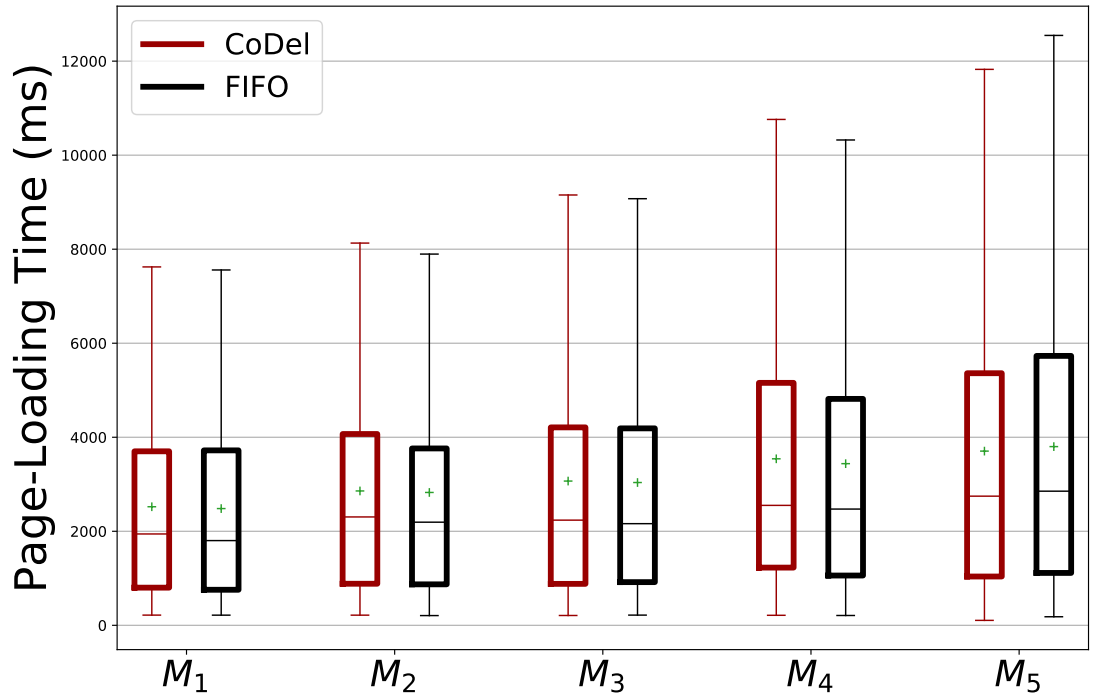
	M_1	M_2	M_3	M_4	M_5
CoDel	8.02	8.19	8.8	9.36	11.83
FIFO	7.94	8.04	8.91	9.47	12.6

LOGISTIC causes higher PLT for most scenarios related to ELASTIC algorithm. This is true regardless of the queue discipline. This result is a consequence of the ELASTIC conservative nature (relying on harmonic bandwidth estimator). This difference is most clear for M_5 case, where PLT increases by $391ms$ and $191ms$ for FIFO and CoDel, respectively.

Overall, both disciplines achieve a similar result across scenarios. For most scenarios FIFO queue achieves lower overall PLT except for the M_5 case where CoDel improves PLT significantly, especially with more aggressive LOGISTIC algorithm, where difference is $305ms$. This comes at the cost of average video QoE, but the impact is less significant. Both disciplines achieve high fairness for video clients (> 0.95) across all scenarios. Percentage of abandonment rates (ABR) for web pages are similar, with FIFO having a slightly higher percentage (still ABR is around 1% on average).



(a) LOGISTIC



(b) ELASTIC

Figure 7.2: Comparison of PLT between CoDel and FIFO Queuing Discipline in two-queue discipline (equal bandwidth share)

While the FIFO queue is the simplest queuing discipline, configuring queue length to 2xBandwidth Delay Product (BDP), capacity, and round-trip-time (RTT) are needed in advance. CoDel operates only by tracking local queuing delay its configuration parameters do not depend on RTT and bandwidth capacity as authors of CoDel claim that default configuration parameters show good results across a wide range of RTT and capacity combinations.

For ENDUE's design, CoDel is selected as the main queuing discipline. Next, CoDel needs to be modified to support larger queuing delays for video traffic. While the initial design of CoDel revolves around keeping low queuing delay, this behaviour needs to be modified for video traffic. The next section analyses CoDel parameters and its impact on web and video performance.

7.2.1.1 Tuning queue parameters

CoDel maintains two parameters: *target* and *interval*. By default, values for these parameters are $5ms$ and $100ms$, respectively. CoDel works by monitoring per-packet latency inside *interval*. If the latency is larger than *target* for at least one *interval*, CoDel starts dropping packets until latency goes down below *target*. For tuning, CoDel's target value is the most important parameter. The main motivation for tuning this parameter lies from the observation in the previous chapter, where added delay improve video QoE. To apply this to ENDUE, the target value needs to be increased allowing higher delay threshold for video traffic. While the authors [NJ12] of CoDel argue that increasing *target* value does not significantly increase utilisation, their observation is made for a one queue discipline only.

Similar to the previous section, evaluation is performed with equal quantum values for both queues, where each queue discipline running CoDel. Web queue is configured with CoDel's default parameters ($5ms$ for target), while multiple target values, ranging from $5ms$ to $40ms$, are tested for the video queue. Scenarios $M_1 - M_5$ are tested with LOGISTIC and ELASTIC adaptation algorithms.

Table 7.4 shows average video QoE for LOGISTIC and ELASTIC algorithms across different M_i scenarios. For M_1 and M_2 case and LOGISTIC algorithm, video performances are similar across different *target* values. Higher *target* values result in a slight increase in the number of switches for video clients, which increases the switching penalty of QoE models and thus reduces its overall score. A similar observation holds for ELASTIC. For scenarios $M_3 - M_5$, video QoE

improvement is more evident as *target* values increase, for both adaptation algorithms.

Table 7.4: Average video QoE score across various *target* values and M_i scenarios with LOGISTIC and ELASTIC adaptation algorithms

LOGISTIC

	M_1	M_2	M_3	M_4	M_5
T5	8.62	8.59	8.95	10.0	11.51
T10	8.61	8.54	8.95	10.04	11.55
T20	8.62	8.55	9.01	10.13	11.64
T40	8.57	8.55	9.05	10.17	11.74

ELASTIC

	M_1	M_2	M_3	M_4	M_5
T5	8.02	8.19	8.8	9.36	11.83
T10	7.97	8.13	8.67	9.26	11.73
T20	8.01	8.12	8.58	8.98	12.31
T40	8.08	8.23	8.95	9.36	12.85

Figure 7.3 depicts boxplots of PLT for LOGISTIC and ELASTIC adaptation algorithms across different M_i scenarios and target values. Performance is similar across all target values and scenarios for both algorithms.

The $20ms$ *target* value strikes a balance in video QoE - PLT tradeoff across all scenarios. In the rest of the experiments, $20ms$ is used for the video queue *target*.

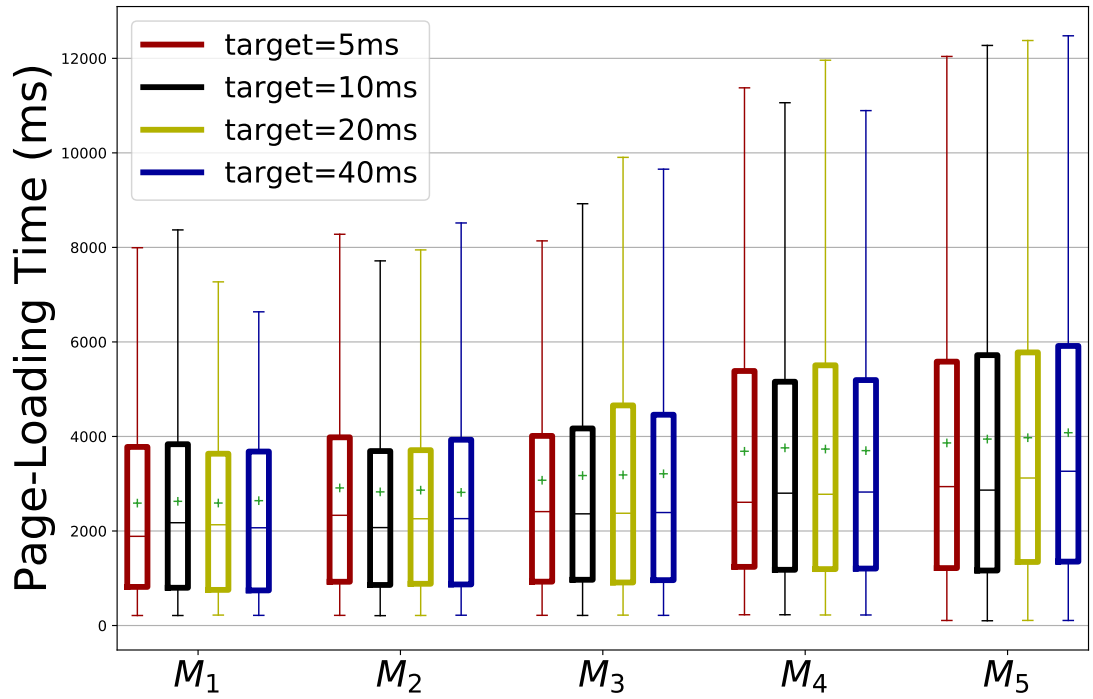
7.2.2 Delay-based adaptive scheduling

After selecting and tuning queues for storing video and web packets, the next step is designing the scheduling logic. ENDUE extends DRR with adaptive quantum values logic. Quantums are updated depending on average queuing delay for both classes. Procedure 1 defines the quantum update logic.

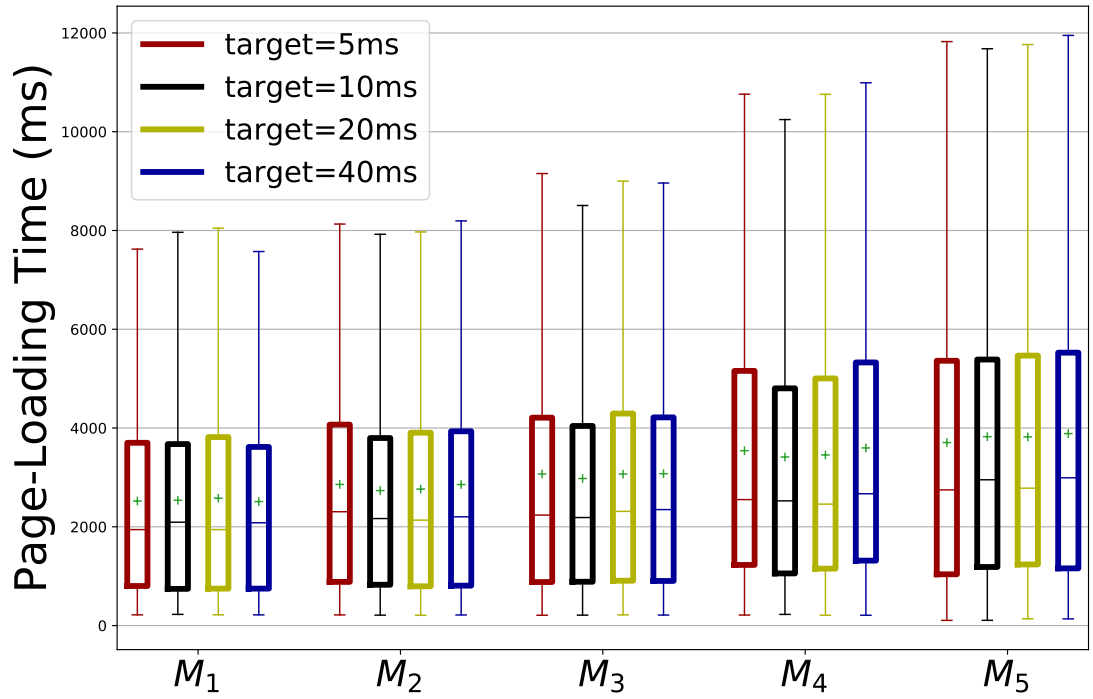
Average delay is calculated using Little's law [PNP+13]:

$$d = \frac{q_{len}}{drate_{avg}} \tag{7.1}$$

where q_{len} is queue length in bits, and $drate_{avg}$ is the average draining rate of the queue. The average draining rate is calculated using Exponential Weighted Moving Average (EWMA) bandwidth estimator with $30ms$ interval. Expression for $drate_{avg}$ is [PNP+13]:



(a) LOGISTIC



(b) ELASTIC

Figure 7.3: Comparison of PLT for different CoDel's target values in two-queue discipline (equal bandwidth share)

Algorithm 1 Procedure for quantum update

repeat(every update *interval ms*)
  Calculate: d_{web}, d_{video} ▷ Calculation of average delay for web and video queues
  $r_{web/video} = \frac{d_{web}+20ms}{d_{video}}$ ▷ Ratio between web and video delay
  if $r_{web/video} > 1$ **then** ▷ Add priority to web traffic
    $q_{web} = MTU \times r_{web/video}$
    $q_{video} = MTU$
  else if $r_{web/video} < 1$ **then** ▷ Add priority to video traffic
    $q_{web} = MTU$
    $q_{video} = MTU \times r_{web/video}$
  else
    $q_{web} = MTU$
    $q_{video} = MTU$
until there are packets backlogged

$$drate_{avg} = (1 - \alpha) \times drate_{avg} + drate \tag{7.2}$$

where $\alpha = 0.125$ is averaging parameter, and $drate$ is rate calculated for the last $30ms$ interval:

$$drate = qlen_{30ms}/(30 \times 10^{-3}) \tag{7.3}$$

where $qlen_{30ms}$ is queue length over last $30ms$.

The key idea is to update the quantum for each class based on average delay. In the results, both queues have the same *default target* value ($5ms$). As video clients are delay tolerant, the video queue should have a larger *target* value than a web queue, allowing increased delay by decreasing the packet-loss rate. Based on the analysis in the previous section, $20ms$ is chosen as *target* value for the video queue. To accompany increased delay for the video queue, Procedure 1 "adds" $20ms$ of virtual delay to web queuing delay to match higher delay threshold of video traffic. The reason is threefold. First, this ensures that actual web delay is always relatively lower than video delay, and it does not depend on link bandwidth. Second, the video queue is configured to have a higher delay ($20ms$ *target*), so $20ms$ of added virtual delay to web queue mirrors that. Third, keeping web queuing delay artificially higher than video dampens maximum *quantum* allocated to video class and thus limits additional delay added to web class while waiting for video class to send all packets in one round.

Next, does this discipline protect video flow from starving? In a case with a high load of web traffic and the low load of video traffic adaptive discipline would starve

video traffic. However, underlying CoDel queues will indirectly manage this delay by increasing packet dropping probability, causing a decrease in web delay and thus a decrease in web quantum. Still, another parameter plays an essential role. How often quantum values are updated can counter above observations.

The following experiments are performed. Five web clients are sharing bandwidth with one video client (M_5). This scenario mimics high web load and relatively low video load. Five update intervals are tested: $25ms$, $50ms$, $100ms$ and $1000ms$. For completeness, similar experiments are repeated for scenarios M_1 and M_3 representing cases with high video load and with equal web and video load scenarios (note that this is not entirely true as HAS can adjust load during playback to adapt to network conditions, but it does not impend observations). Note that, Table 7.5 depicts average bandwidth allocation calculated from assigned quantum values during the experiment.

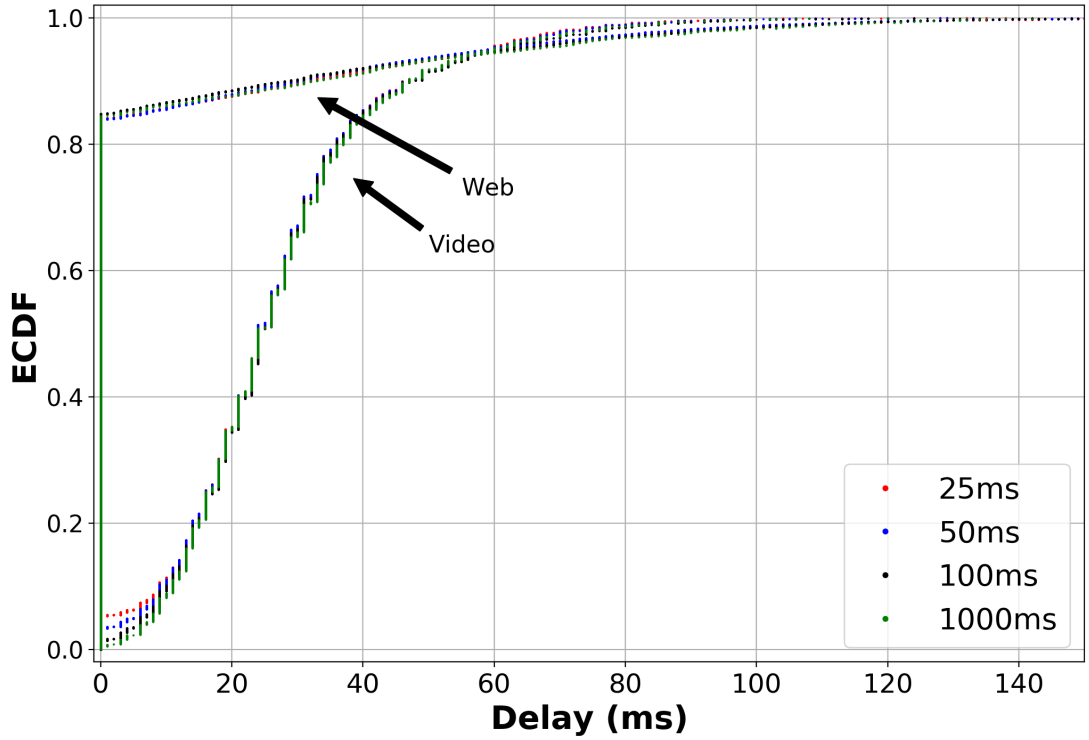


Figure 7.4: ECDF of average queuing delay for different update intervals (M_1)

Figure 7.4 depicts Cumulative Distribution Function (CDF) of web and video average queuing delay for the M_1 scenario. For 80% of the time, the web queuing delay is near zero. This result is not surprising because of the low web load (one web client). The allocated bandwidth share is 9% on average, as illustrated in Table 7.5a. For video traffic, queuing delay is less than $37ms$ for 80% of the time, resulting in 90% allocated bandwidth share. Different update intervals do

not have a significant impact on performance. This result is expected, as the discipline is biased towards web traffic. Overall, the discipline achieves the main goal: web queuing delay is lower than video queuing delay.

For M_3 scenario web delay increases with 60% of time delay being near zero, with video delay equal to $25ms$ across all update intervals. The increase in delay is followed by higher bandwidth share, as illustrated in Table 7.5b. Web traffic receives 27% of bandwidth share on average.

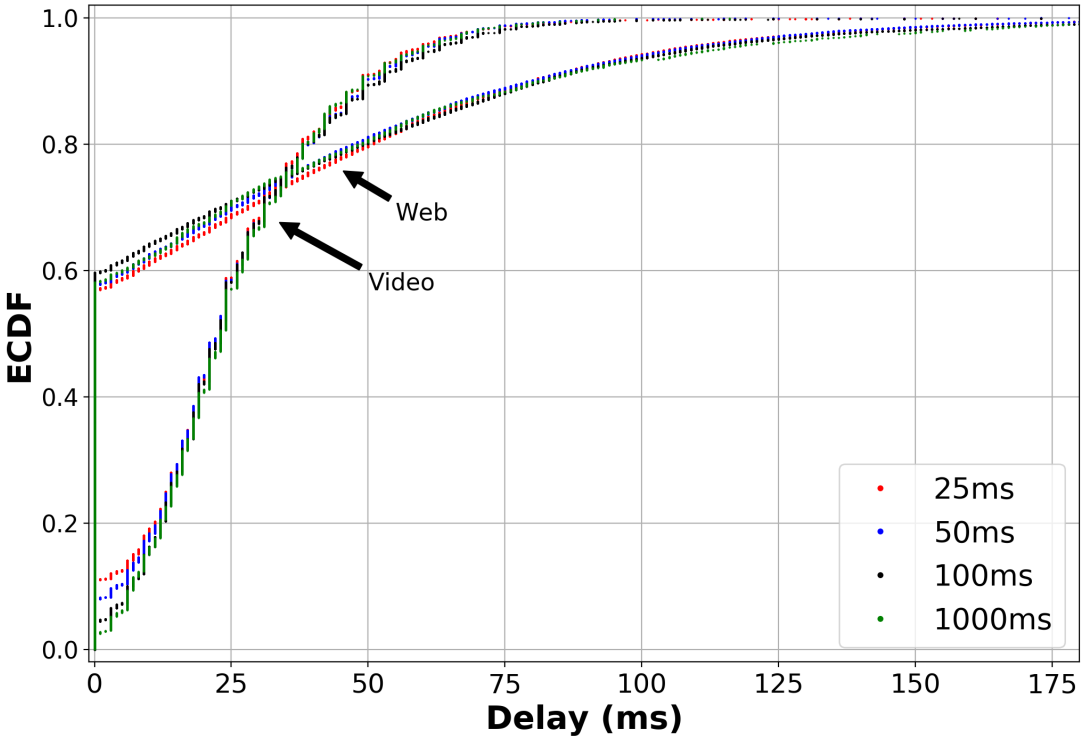


Figure 7.5: ECDF of average queuing delay for different update intervals (M_3)

Finally, Figure 7.6 depicts an empirical CDF for web and video average queuing delay for M_5 across different update intervals. The delay for web traffic is higher than the delay for video traffic. This is the expected result, as only one video client is competing with five web clients. Increasing the update interval decreases delay for both queues.

First, the average allocated bandwidth is much higher than fair share calculated solely based on the number of clients (this is the approach taken FQ-CoDel). There is no significant difference across different intervals. Average offered load of the web client is 890Kbps. On average, the scheduler will allocate 3.6Mbps for web queue or 4x client average load. This will lead to increased latency and higher packet loss for web queue as the outgoing capacity can not support all five

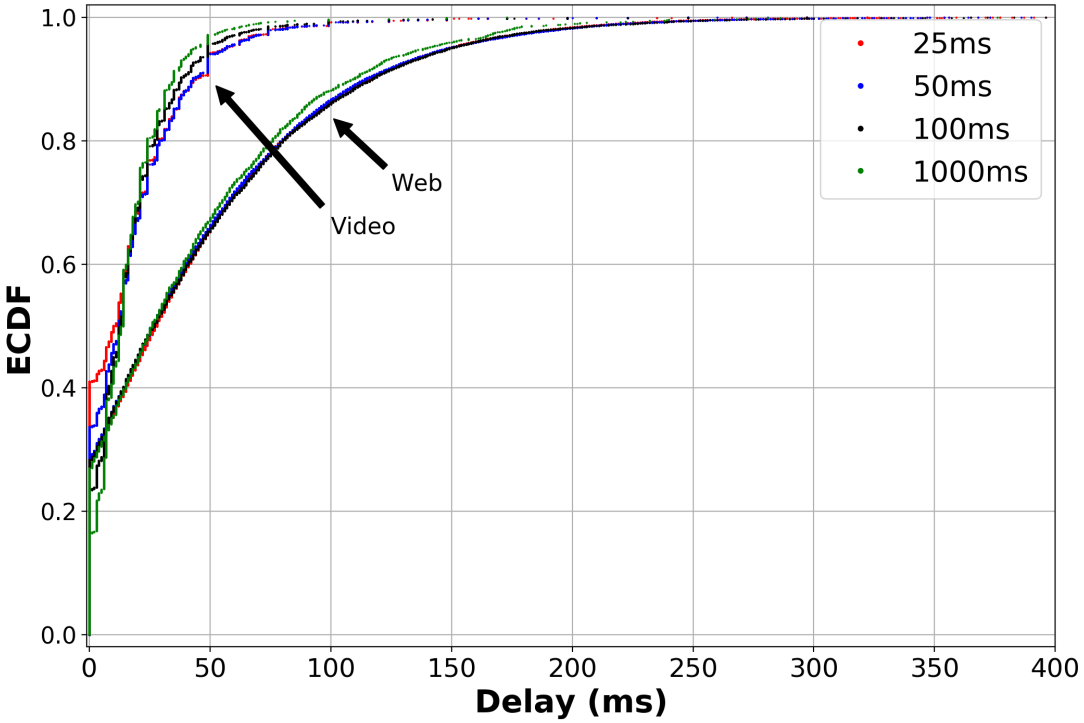


Figure 7.6: ECDF of average queuing delay for different update intervals (M_5)

Table 7.5: Average bandwidth allocation for web and video flows across different update intervals and M_i scenarios

(a) M_1

Interval	Web	Video
$25ms$	9%	91%
$50ms$	9%	91%
$100ms$	9%	91%
$1000ms$	9%	91%

(b) M_3

Web	Video
27%	73%
27%	73%
26%	74%
28%	72%

(c) M_5

Web	Video
57%	43%
57%	43%
57%	43%
58%	42%

clients at the same time. This observation can be confirmed looking at packet loss for each queue. For web queue, average packet loss is 1663 ppr (packets per run) while packet loss for video queue is 330 ppr. Packet loss for web queue is 5x higher than for video queue, expected result as web load is higher than video load, and the queue web has lower *target* threshold. At the application level for M_3 and M_5 scenarios, performance unravels a clear pattern (not shown). Increasing the update interval decreases overall video QoE while improving web page-loading time. This decrease in video QoE is the consequence of the increased number of switches (up to 20%) and decreased average bitrate quality.

As a tradeoff, the update interval of $100ms$ is selected. In any case, the proposed discipline will not starve video traffic even in scenarios with high web load.

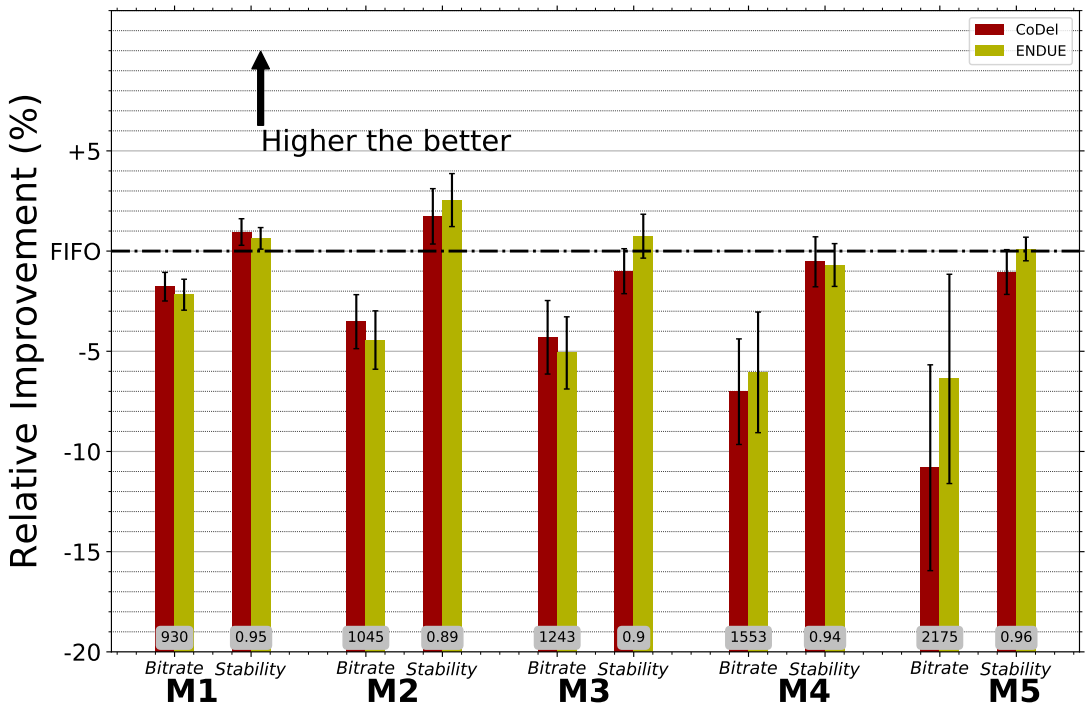
7.3 FIFO vs CoDel vs ENDUE

After completing the design of the ENDUE queuing discipline, this section presents the results of a comparison between the proposed discipline and CoDel. Also, all experiments are evaluated against the FIFO queuing discipline (2xBDP) as the base result.

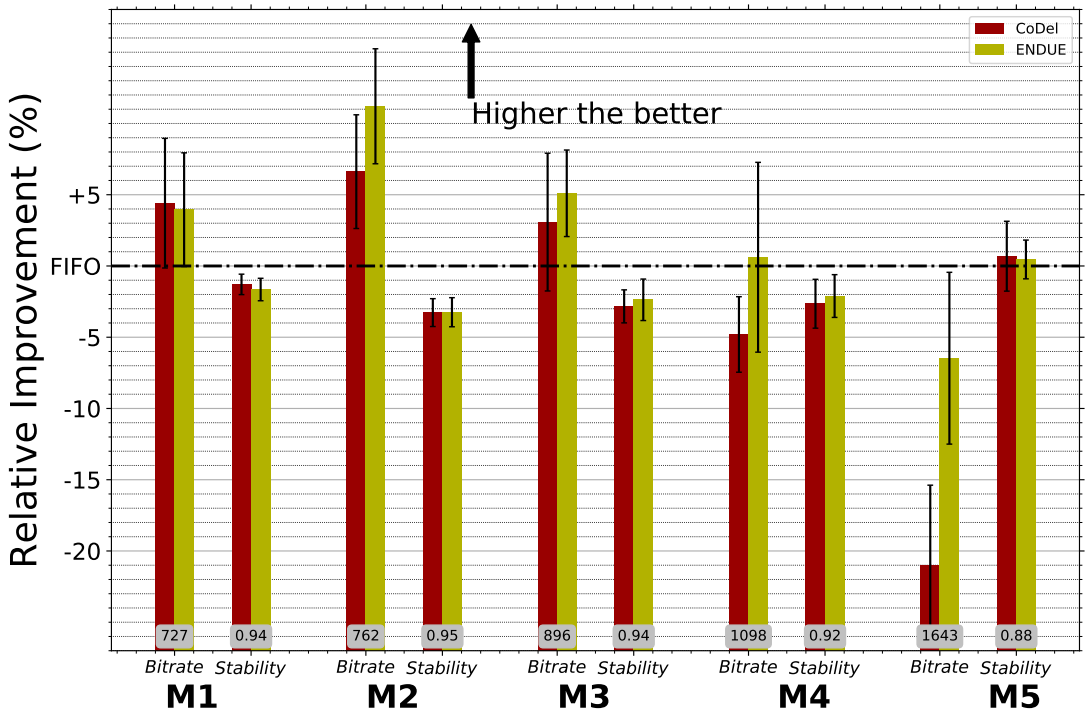
Figure 7.7 depicts video QoE metrics, average bitrate and stability performance, across different combinations of web and video clients for CoDel and ENDUE. For the LOGISTIC algorithm, there is a negligible difference between CoDel and ENDUE across different M_i scenarios for both QoE metrics. For the stability metric, the difference is around 1% on average. In the M_5 case (five web and one video clients) average bitrate shows the highest difference, 5%. Compared with the FIFO queuing discipline, both CoDel and ENDUE achieve a lower average bitrate. This difference increases as the number of video clients decreases. However, with ENDUE, the difference is less than 5% for all cases. In the M_5 case, CoDel decreases average bitrate by 11%. For the stability metric, performance is similar across all three disciplines.

For the ELASTIC algorithm, the stability metric is similar across all queuing disciplines and M_i scenarios. For average bitrate, CoDel and ENDUE achieve higher bitrate for scenarios $M_1 - M_3$ than FIFO. Still, difference is negligible in all cases, expect for M_5 scenario in which bitrate drops by 20% for CoDel compared to FIFO and ENDUE. Unlike LOGISTIC algorithm, ELASTIC, similar to many algorithms, takes rate estimation when deciding for the next chunk quality. Also, ELASTIC does not use any gradual rate step up. This causes the issue with one video client sharing a bottleneck with multiple web clients (M_5 case). The video client overestimates its bandwidth share and starts streaming at the highest bitrate quality. However, as web clients load webpages, download time of chunk drastically increases and together with low buffer level causes the player to stall (on the average CoDel produces 1.4 seconds of stall, FIFO 0.67 seconds, and ENDUE 0.87 seconds). This limits overall bitrate and explains the vast difference between CoDel, ENDUE and FIFO discipline. Added delay for video traffic class counter this startup behaviour and thus significantly improves bitrate.

Table 7.6 depicts F-index for LOGISTIC adaptation algorithm across different M_i scenarios. All queuing disciplines show the same values across all scenarios. Clients achieve an almost perfect share in terms of QoE. This is a consequence



(a) LOGISTIC



(b) ELASTIC

Figure 7.7: Video performance in presence of web clients for CoDel and ENDUE queuing discipline (*The metrics are normalised to the FIFO case with numbers in white boxes representing the value of each metric for the FIFO case*))

Table 7.6: F-Index across various M_i scenarios with LOGISTIC adaptation algorithm (M_5 is omitted as it only has one video client competing for resources)

	M_1	M_2	M_3	M_4
CoDel	0.99	0.99	0.99	0.99
ENDUE	0.99	0.99	0.99	0.99
FIFO	0.99	0.99	0.99	0.99

of the LOGISTIC algorithm design. LOGISTIC is a buffer based algorithm, and decision for next chunk quality is only depended on the buffer level.

For the ELASTIC algorithm, F-Index is significantly susceptible to bursty web traffic, as illustrated in Table 7.7. Increasing the number of web clients negatively affects F-Index, dropping to 0.8 for M_4 scenario and CoDel discipline. Similar observations hold for FIFO, but effect is less pronounced because of higher queue length (2xBDP) and thus higher queuing delay, which improves fairness. Traffic isolation and added delay of ENDUE discipline helps in alleviating this issue, achieving constant high F-index across different scenarios.

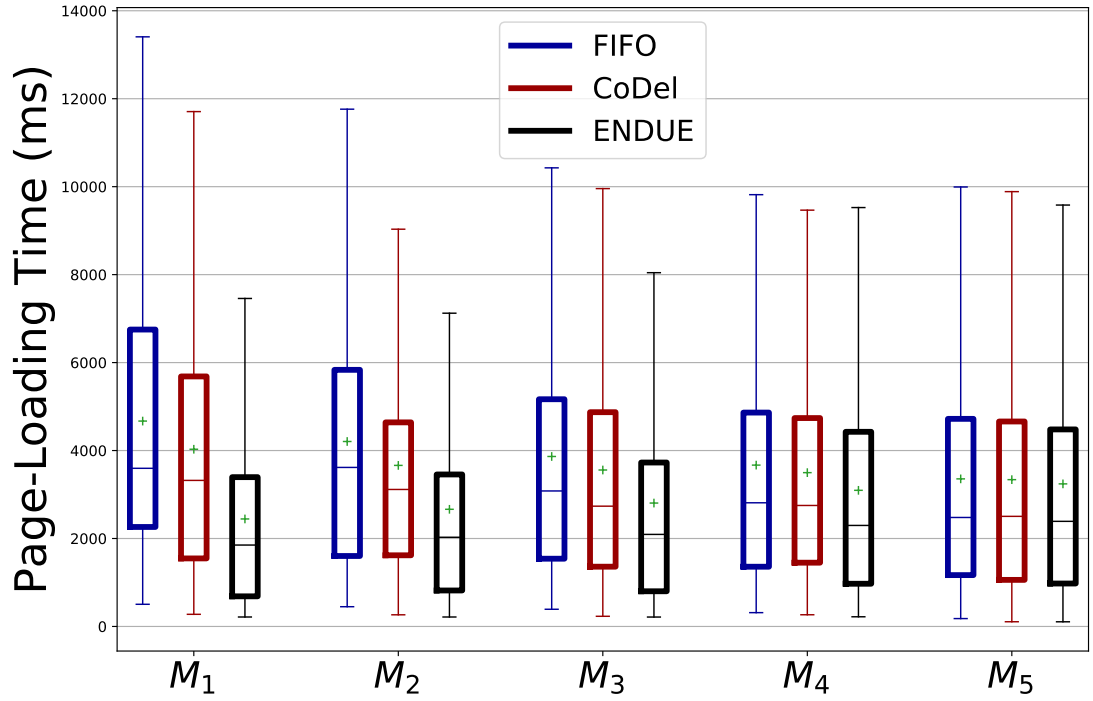
Table 7.7: F-Index across various M_i scenarios with ELASTIC adaptation algorithm (M_5 is omitted as it only has one video client competing for resources)

	M_1	M_2	M_3	M_4
CoDel	0.97	0.92	0.88	0.8
ENDUE	0.98	0.98	0.98	0.98
FIFO	0.94	0.96	0.94	0.91

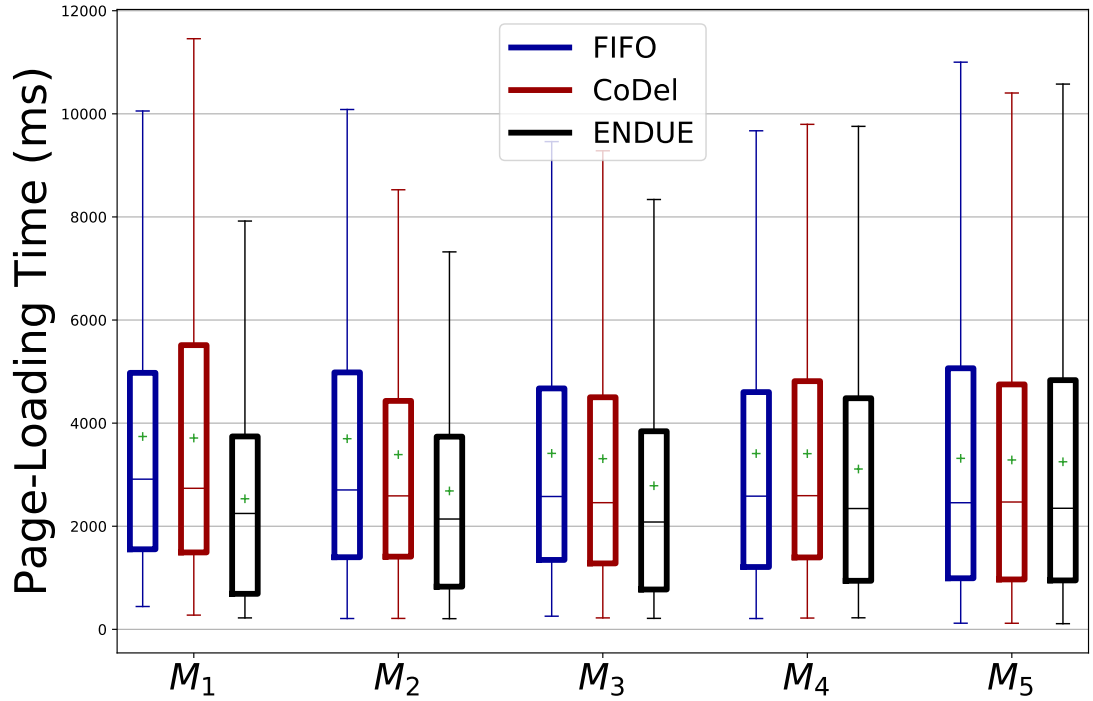
Next, web performance is evaluated. Figure 7.8a shows PLT across different M_i scenarios for LOGISTIC adaptation algorithm and FIFO, CoDel and ENDUE discipline.

Increasing the number of video clients harm PLT when CoDel queuing discipline is used. Median PLT increases by the significant 33% from M_5 to M_1 scenario. However, for a ENDUE, median PLT drops leading to improved PLT with no significant impact on video QoE metrics, thanks to extended queuing. In all scenarios, PLT is lower compared to the same scenario with CoDel or FIFO. Also, for the M_1 scenario, median PLT is 94% and 80% lower compared to the FIFO and CoDel disciplines, respectively.

By its design, ELASTIC adaptation algorithm is more conservative compared to LOGISTIC. As a result, its impact is less pronounced on PLT as illustrated in Figure 7.8b. For FIFO and CoDel, the difference in median PLT values across different scenarios is 19% and 11% respectively. Overall, PLT is smaller compared



(a) LOGISTIC



(b) ELASTIC

Figure 7.8: Comparison of PLT between CoDel and FIFO Queuing Discipline in two-queue discipline

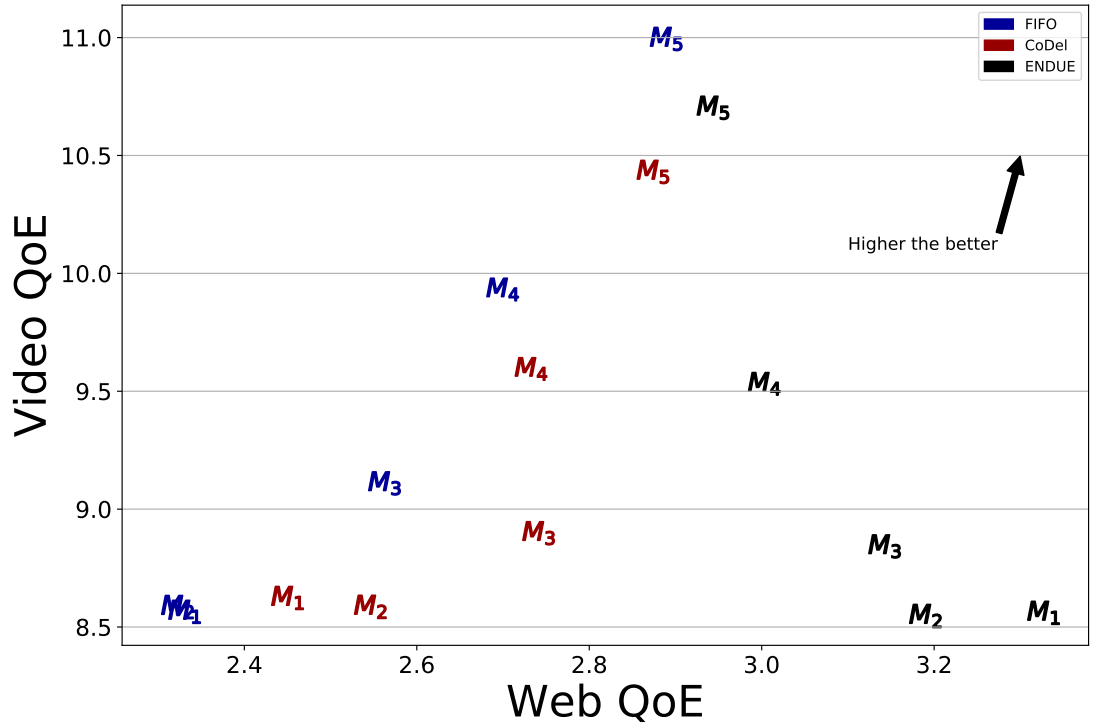
to the same scenarios with the LOGISTIC algorithm. Still, the ENDUE discipline improves PLT in all cases, while having an insignificant impact on video QoE metrics.

Similar abandonment ratio is achieved across all scenarios and adaptation algorithms for CoDel and ENDUE. On average, this ratio is less than 2%. Considering that median PLT is around 2 seconds on average, this result is not surprising as the threshold for the page abandonment is 15 seconds [TZS14]. FIFO shows a higher abandonment rate, on average, 3% across all scenarios and algorithms. This result is intuitive FIFO does not use any random dropping techniques for packets causing higher queuing delays for both traffic types.

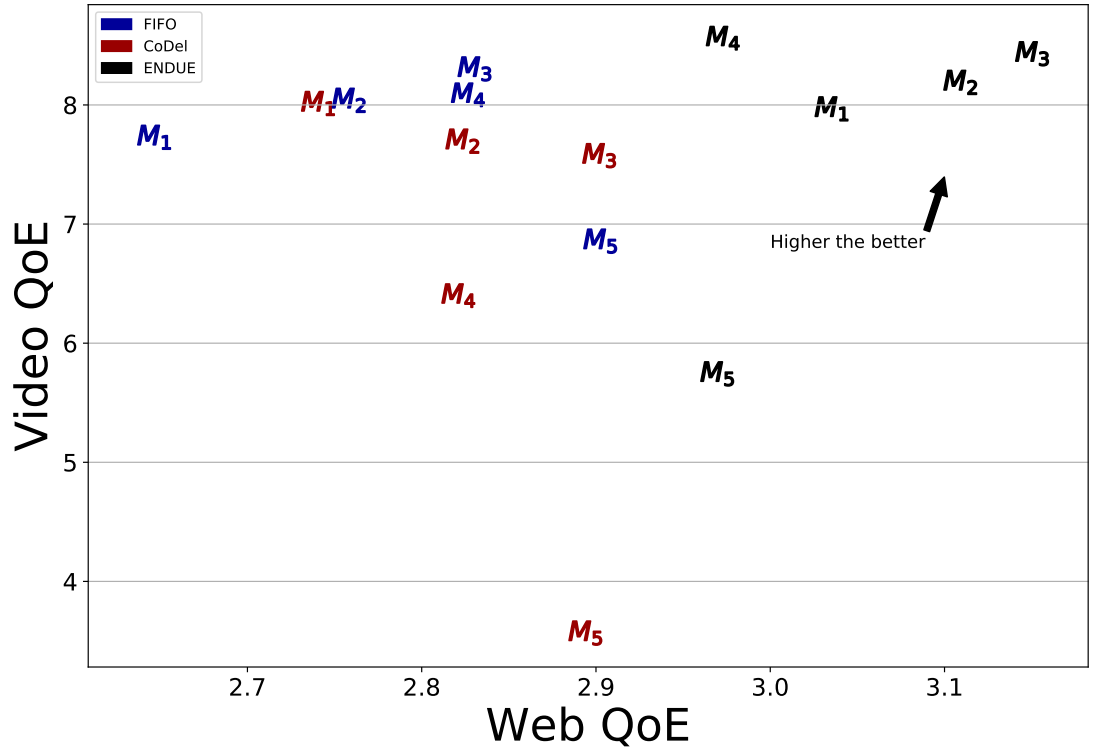
Figure 7.9 depicts a scatter plot for video and web QoE. In the LOGISTIC's case adaptation algorithm, ENDUE improves web QoE significantly in most scenarios (up to 40%), while having a negligible negative impact on video QoE. Similar conclusion holds for ELASTIC except in M_5 case, where deterioration in stall performances harms overall video QoE. As already explained this is consequence of ELASTIC design. Still, ENDUE improves web QoE up to 13%.

7.4 Discussion

Filtering web and video traffic can pose a challenge as the majority of traffic is encrypted. However, one approach is to identify Server Name Identification (SNI) from Hypertext Transfer Protocol Secure (HTTPS) encrypted traffic to identify content provider and content delivery network [MZA+18]. Content filtering is a common practice applied by most network operators. Many operators allow an unlimited data plan for services such as YouTube and similar. Implementing ENDUE in real routers is straightforward. All the experiments performed in this chapter are based on real-time and real traffic evaluation. ENDUE is implemented as part of the *tc* library, similar to CoDel queuing discipline. In terms of running times, ENDUE uses simple arithmetic operations. Also, modified deficit round robin uses MTU as the minimum quantum, ensuring efficient packet processing. The proposed discipline scales with the increased amount of traffic. ENDUE logic doesn't directly depend on the number of competing flows. This is because ENDUE only tracks the local queuing delay for both queues and update weight based on relative difference.



(a) LOGISTIC



(b) ELASTIC

Figure 7.9: Scatter plot of video and web QoE across multiple scenarios, queuing disciplines and adaptation algorithms

7.5 Conclusion

This chapter introduced ENDUE a simple two-queue discipline with delay-based resource allocation for the heterogeneous traffic sharing bottleneck link. Heterogeneous traffic consists of two traffic types: video (streamed using HTTP adaptive concepts) and web browsing. These two types are the most popular traffic carried over the Internet.

The ENDUE discipline takes application characteristics into account when scheduling packets. As a result, compared to the state-of-the-art AQM discipline, CoDel, ENDUE decreases median PLT of web traffic by up to 80% compared to CoDel. This results in up to 27% web QoE improvement. Also, there is no significant negative impact on video QoE regardless of the scenario and adaptation algorithms (video QoE drops by less than 1% compared to CoDel).

Chapter 8

Conclusion

The achievable quality of video streamed over the Internet is affected by the limited network resources (i.e. available bandwidth) shared among multiple clients. HTTP adaptive streaming (HAS) enables seamless adaptation of video quality to changes in the available throughput.

This is achieved by splitting video content into multiple several seconds (typically 2-10 seconds) chunks. Each chunk is encoded into multiple qualities allowing multiple bitrates and resolutions per chunk. As each chunk can be decoded independently of other chunks, changing the qualities per chunk is possible seamlessly. Many HAS algorithms were designed over the years to maximise streamed quality while minimising stalls and switching between different qualities.

All the HAS algorithms base their decision on the measurement of the available throughput. There are two ways to get this information, directly by measuring chunk download time and indirectly by tracking playback buffer levels. Intuitively, the accuracy of these measurements has the most impact on HAS performance.

However, obtaining throughput estimates is a challenging task, especially in highly variable environments, such as cellular networks. The achievable throughput for devices in cellular networks can fluctuate by an order of magnitude over a span of a few seconds for a variety of reasons. There can be rapid changes in the underlying radio channel conditions and system load as devices move and new devices enter and existing devices leave the network.

In addition, system-specific factors such as observed network delay, network capacity, the characteristics of competing traffic, and the traffic management policy

significantly impact the HAS performance. The combination of the relatively high data rates of individual video flows and the popularity of video streaming led to interest in analysis of the performance when multiple video flows compete for network resources. Furthermore, specific concerns evolved regarding performance-related interactions between HAS and non-HAS traffic.

Hence, this thesis focuses on improving throughput estimation methods in highly variable environments such as cellular networks by utilising additional information about channel characteristics. Also, interactions between multiple HAS and non-HAS clients are analysed and the increase in their performance is sought by improving the network scheduling discipline.

8.1 Contributions

The first contribution includes the analysis of state-of-the-art HAS algorithms in the presence of ideal throughput prediction. Leveraging the modular design of HAS algorithms enabled improving Quality of Experience (QoE) regardless of algorithm design. Different ways to feed throughput prediction values to algorithms were explored together with different prediction horizons. Regardless of the algorithm in use, user QoE improves by a significant 23% in the presence of accurate throughput prediction. Furthermore, the highest QoE is observed in the presence of longer throughput prediction horizons. Most notably, accurate prediction eliminates stall events in an environment with highly fluctuating throughput. While error-induced prediction lowers significantly the user QoE in some instances, it still provides a clear 15% gap on average, compared to HAS algorithms with no prediction.

Motivated by these findings, a novel Machine Learning (ML)-based throughput prediction technique is developed. This technique leverages radio metrics to improve the throughput prediction accuracy in mobile networks significantly. Combining machine learning techniques with radio channel metrics summarised by a novel quantile abstraction technique enables achieving low throughput prediction errors (90% of errors below 13%). Utilising this abstraction technique enables capture of trends and variation in metric data accurately in the environment where metrics are updated/available at a fine time granularity. All tested algorithms improve all QoE metrics when using realistic prediction. Notably, prediction reduces stalls by up to 85%, and bitrate switching by up to 40%, while maintaining or improving video quality. As a result, the QoE score improves significantly by

up to 33%. Finally, a prediction framework is implemented in mobile devices and field evaluation is performed in an operational cellular network.

The second contribution is the exhaustive empirical study. The results from this study show that the well-known "rule-of-thumb" (1xBandwidth Delay Product (BDP)) for network queue dimensioning causes underutilisation (70%) when multiple HAS clients share a bottleneck. Larger queues, e.g. 2xBDP, can improve utilisation and quality by 15% on average. With median queue size, overall QoE improves by 12% on average. Larger queues also help in improving system fairness for clients with heterogeneous round-trip-times (RTTs). However, larger queues can negatively impact delay-sensitive traffic, causing *bufferbloat* phenomenon. While modern Active Queue Management (AQM) techniques promise low delay and high utilisation and are application-agnostic, our results show that their performance is mostly scenario-dependant and would vary depending on bitrate distribution, video adaptation algorithm and offered web traffic load.

Motivated by these results, third contribution is the design of an effective two-queue discipline for traffic scheduling. The proposed queuing discipline takes application characteristics into account when scheduling packets. As a result, compared to the state-of-the-art AQM discipline, Controlled Delay Management (CoDel), the proposed discipline decreases median Page Loading Time (PLT) of web traffic by up to 80% compared to CoDel. This results in up to 27% web QoE improvement. Also, there is no significant negative impact on video QoE regardless of the scenario and adaptation algorithms (video QoE drops by less than 1% compared to CoDel). This is preceded by an exhaustive evaluation of the interactions between HAS and non-HAS traffic in the constrained environment. This analysis enabled drawing observations that helped in designing the two-queue scheme.

As the part of analysis, tools and techniques were developed that allow gathering and analysing application related data and performing all HAS-related experiments. As a result, a 4G dataset was produced with unique information (such as the channel, user and throughput information) and has been released to the research community. Also, a testbed framework for performing HAS-related experiments has been released to the research community.

8.2 Future work

8.2.1 Extending throughput prediction analysis

In the experiments, the bandwidth estimator is simply replaced with a prediction value and constant horizon. However, the optimal horizon is unknown in advance. Furthermore, none of the tested algorithms is optimised to take full benefit from more accurate predictions. In previous studies [MTAA+16, ZEG+15], authors show that HAS algorithms can be fully tuned based on the prediction horizon. Furthermore, some of the algorithms take multiple horizons into account when deciding the next quality. Arguably, having various horizons, i.e. short and long horizon, can improve user experience even further than only taking one constant horizon. However, generating multiple horizons in real-time is challenging. Each horizon would require a separate prediction model. As a result, different sets of training data are needed for each horizon and thus training appropriate ML model per horizon. Storing and running these prediction models can pose a challenge on power-constrained devices (e.g. smartphones). Designing a prediction-aware adaptation algorithm is a natural next direction for improving video QoE beyond current results.

The implementation of device-based or network-based implies a different set of challenges for every design choice. Having a `Predictor` on the device represents a timely and scalable option as the metrics sampling and prediction are collocated in the same device. However, device capabilities can limit prediction accuracy due to limited hardware resources, especially for ML models. Hence, loading and running ML models on mobile devices becomes impractical due to the limited memory and CPU power. One of the solutions is to train the model with a subset of data, hence limiting its size and decreasing its prediction accuracy (Section 5.4). Still, mobile technology is quickly closing the gap in terms of sheer processing power together with dedicated hardware chips for ML tasks as a part of a mobile device's architecture, alleviating this drawback in the near future.

For a network-level prediction system, current cellular networks (4G) have limited support for deploying ML at scale. The lack of a standardised way for data collection from eNodeBs across different vendors, together with restricted access to eNodeBs makes ML integration limited to only a few eNodeBs [PJK+18]. Next-generation cellular networks (5G) offer the opportunity for successful in-network ML deployment. One possible architecture as outlined in [PJK+18]

proposes deploying ML applications in the Mobile Edge Cloud and leveraging network controllers for data collection from Next Generation Node Base (ngNodeBs). Network controllers govern over multiple ngNodeBs striking a balance between distributed (e.g. 4G) and a centralised architecture (e.g. one controller controlling all ngNodeBs). Striking the right balance between the number of user sessions, amount of traffic and the network state view per network controller allows for this architecture to scale over millions of devices. Having an ML model in the edge minimises delivery latency to possible sub-millisecond values.

The implementation of ML-based solutions in real systems is known to bring new challenges that need further exploration. "Dataset shift" [MTRAR+12] is a common problem when training and test data come from two different distributions. This phenomenon may occur in "non-stationary environments" where the training environment is different from the test one. Another possible cause is training ML model with data that does not fully represent various operational contexts, e.g. training using only mobile users that may not fully capture behaviour of static users. Hence, the performance of throughput prediction in real-networks still needs exploration, possibly using techniques such as transfer and active learning.

The interplay between application traffic and metrics collection poses challenges. Specifically, the measured throughput, which is used both as a model feature and ground truth for training, is naturally affected by the application traffic pattern. Existing studies rely on persistent traffic that saturates the link (e.g. downloading a large file) to probe the available capacity. However, many applications download relatively small files that may not be sufficient to probe the available system resources. An example is video streaming where the user downloads low video quality segments (a few hundred Kilobytes) that may lead to lower throughput metrics [HQG+13]. Noting that physical-layer metrics are independent of the application, the pattern offered to the predictor may produce inaccurate predictions in such cases.

The application activity is another factor that may impact the availability and fidelity of throughput metrics. An inactive application lacks historic throughput information. Another prominent case is when application activity is smaller than the sampling interval. Handling these situations still requires further research. One possible approach is to integrate application-specific metrics as part of the throughput prediction model.

The application using throughput prediction would benefit from knowing how much confidence to put in the prediction. For example, a video application may

act conservatively if buffer occupancy is low and throughput prediction has low confidence. Identifying these scenarios also gives us motivation to develop alternative prediction techniques to handle these problem situations.

In the experiments, predictions are calculated at the device itself. While this approach brings benefits regarding scalability and possibility to retrain the model to suit better for local conditions, its prediction power can be limited for reasons outlined in previous sections. Unlike device-based prediction, in-network prediction does not suffer from limited computational power. The device can send its metrics periodically and request forecast when needed. However, this implies having close coordination and co-operation between end-device and network provider/service.

8.2.2 Further analysis of the network queue impact on user experience

The proposed ENDUE discipline shows promising results. As future work, more theoretical analysis would be beneficial for better understanding and controlling queuing delay. One approach is to use fluid modelling [MGT00] and a derived fluid model for the proposed discipline. For this, the fluid model of CoDel is needed. Recently, Patil et al. [PT19] derived a fluid model for CoDel, enabling a detailed analysis of the proposed discipline.

While the ENDUE discipline uses a simple ratio for adaptation of quantum values, a more sophisticated control discipline Proportional Integral Derivative (PID) may give benefits. While PID is simple enough, it requires considerable effort to tune parameters. However, in the literature, a PID control law has been used for designing AQM techniques [XHY+05, PNP+13, DSBTB16]. For PID tuning, a theoretical model is needed, as explained in the previous paragraph.

Chapters 6 and 7 examine multiple heterogeneous traffic user sharing network resources. This scenario reflects home access networks. However, it would be interesting to repeat the same analysis and proposed a solution in the core network part, with a much larger number of users and traffic volume. In the scenarios with over 100 competing users, Transmission Control Protocol (TCP) connections become de-synchronised [AKM04] (i.e. flows do not experience packet drops at the same time).

8.3 Summary

Today, HAS is the most dominant technique for delivering video content. HAS enables changing video quality seamlessly. Simplicity and low cost compared to traditional solutions enabled the adoption of HAS across heterogeneous networks and devices. Still, performance of HAS are desired in environments with frequent throughput fluctuations. The rise of HAS traffic has led to performance issues in scenarios when different applications, such as web traffic (delay sensitive), share network resources with HAS clients. This thesis explores ways to improve throughput prediction in cellular networks through the combination of network-related information about channel and ML algorithms to improve HAS QoE. Also, analysis of interactions between HAS and non-HAS clients are performed to improve QoE of both types of traffic through better network queue scheduling.

References

[AABB19] R. Al-Saadi, G. Armitage, J. But, and P. Branch. A survey of delay-based and hybrid tcp congestion control algorithms. *IEEE Communications Surveys Tutorials*, 21(4):3609–3638, Fourthquarter 2019.

[AABD12] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine 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 '12, pages 9–14. ACM, 2012.

[Ada13] R. Adams. Active Queue Management: A Survey. *IEEE Communications Surveys Tutorials*, 15(3):1425–1476, Third 2013.

[AKM04] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing Router Buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4):281–292, August 2004.

[ALH+18] Eneko Atxutegi, Fidel Liberal, Habtegebreil Kassaye Haile, Karl-Johan Grinnemo, Anna Brunstrom, and Ake Arvidsson. On the use of tcp bbr in cellular networks. *Comm. Mag.*, 56(3):172–179, March 2018.

[ANBD12] Saamer Akhshabi, Sethumadhavan Narayanaswamy, Ali C. Begen, and Constantine Dovrolis. An Experimental Evaluation of Rate-adaptive Video Players over HTTP. *Image Commun.*, 27(4):271–287, April 2012.

[ARhM+17] A. Asan, W. Robitza, I. h. Mkwawa, L. Sun, E. Ifeachor, and A. Raake. Impact of video resolution changes on QoE for adap-

tive video streaming. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 499–504, July 2017.

[BBHZ18] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann. Want to Play DASH?: A Game Theoretic Approach for Adaptive Streaming over HTTP. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys'18. ACM, 2018.

[BBHZ19] Abdelhak Bentaleb, Ali C. Begen, Saad Harous, and Roger Zimmermann. Game of Streaming Players: Is Consensus Viable or an Illusion? *ACM Trans. Multimedia Comput. Commun. Appl.*, 15(2s):65:1–65:30, July 2019.

[BC97] Paul Barford and Mark Crovella. An Architecture for a WWW Workload Generator. In *World Wide Web Consortium Workshop on Workload Characterization*, Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 1997.

[BC98] Paul Barford and Mark Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '98/PERFORMANCE '98, pages 151–160, New York, NY, USA, 1998. ACM.

[BDCR16] Enrico Bocchi, Luca De Cicco, and Dario Rossi. Measuring the Quality of Experience of Web Users. In *Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*, Internet-QoE '16, pages 37–42. ACM, 2016.

[BHK+16] Ayub Bokani, Mahbub Hassan, Salil S. Kanhere, Jun Yao, and Garson Zhong. Comprehensive Mobile Bandwidth Traces from Vehicular Networks. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 44:1–44:6. ACM, 2016.

[BM03] Gustavo Batista and Maria Carolina Monard. A Study of K-Nearest Neighbour as an Imputation Method. In *Hybrid Intell Syst, Ser Front Artif Intell Appl 87, IOS Press*, HIS'03, page 251–260, 2003.

[Bre01] L. Breiman. Random Forests. *Machine Learning*, 45(1), 2001.

[Bri07] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. *SIGCOMM Comput. Commun. Rev.*, 37(2):63–74, March 2007.

[Bru13] Glen Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2013.

[BSJ+11] I. M. Bălan, B. Sas, T. Jansen, I. Moerman, K. Spaey, and P. Demeester. An enhanced weighted performance-based handover parameter optimization algorithm for LTE networks. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):98, Sep 2011.

[BTB+19] 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*, 21(1):562–585, Firstquarter 2019.

[BTBZ19] Abdelhak Bentaleb, Christian Timmerer, Ali C. Begen, and Roger Zimmermann. Bandwidth Prediction in Low-latency Chunked Streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '19, pages 7–13, New York, NY, USA, 2019. ACM.

[CCPM13] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *IEEE International Packet Video Workshop*, Dec 2013.

[Cis17] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021, 2017.

[CMH+18] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz. An Overview of Core Coding Tools in the AV1 Video Codec. In *2018 Picture Coding Symposium (PCS)*, pages 41–45, June 2018.

[CMK+13] Jiasi Chen, Rajesh Mahindra, Mohammad Amir Khojastepour, Sampath Rangarajan, and Mung Chiang. A Scheduling Framework for Adaptive Video Delivery over Cellular Networks. In *Pro-*

ceedings of the 19th Annual International Conference on Mobile Computing & Networking, MobiCom '13, pages 389–400, New York, NY, USA, 2013. ACM.

[CPG+13] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda. Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey. *IEEE Communications Surveys Tutorials*, 15(2):678–700, Second 2013.

[DCCPM14] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. TAPAS: A Tool for rApid Prototyping of Adaptive Streaming Algorithms. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, VideoNext '14, pages 1–6, New York, NY, USA, 2014. ACM.

[DD06] Amogh Dhamdhere and Constantine Dovrolis. Open Issues in Router Buffer Sizing. *SIGCOMM Comput. Commun. Rev.*, 36(1):87–92, January 2006.

[DDR13] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. Model for estimating qoe of video delivered using http adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1288–1293, May 2013.

[DJD05] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested Internet links. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1072–1083 vol. 2, March 2005.

[DSBTB16] Koen De Schepper, Olga Bondarenko, Ing-Jyh Tsang, and Bob Briscoe. PI2: A Linearized AQM for Both Classic and Scalable TCP. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 105–119. ACM, 2016.

[ERHS12] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz. "Time is bandwidth"? Narrowing the gap between subjective time perception and Quality of Experience. In *2012 IEEE International Conference on Communications (ICC)*, pages 1325–1330, June 2012.

[Eri17] Ericsson. Ericsson Mobility Report November 2017, 2017. `https://www.ericsson.com/en/mobility-report/reports/`

november-2017.

[ESS+13] A. E. Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. Steinbach. Quality-of-experience driven adaptive HTTP media delivery. In *2013 IEEE International Conference on Communications (ICC)*, pages 2480–2485, June 2013.

[Eur13] European Union. Quality of Broadband Services in the EU. http://ec.europa.eu/newsroom/dae/document.cfm?action=display&doc_id=10816, 2013. A study prepared for the European Commission DG Communications Networks, Content & Technology.

[FBGP+17] Pedro Fernandes, Marco V. Bernardo, António M. G. Pinheiro, Paulo T. Fiadeiro, and Manuela Pereira. Quality Comparison of the HEVC and VP9 Encoders Performance. *Multimedia Tools Appl.*, 76(11):13633–13649, June 2017.

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug 1993.

[For02] B. A. Forouzan. *TCP/IP Protocol Suite*. McGraw-Hill, Inc., New York, NY, USA, 2 edition, 2002.

[FPT+16] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. An Internet-Wide Analysis of Traffic Policing. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 468–482. ACM, 2016.

[GHFN12] Andrei Gurtov, Tom Henderson, Sally Floyd, and Yoshifumi Nishida. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582, April 2012.

[GN11] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark Buffers in the Internet. *Queue*, 9(11):40:40–40:54, November 2011.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger,

editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[GWZ+09] H. Ge, X. Wen, W. Zheng, Z. Lu, and B. Wang. A History-Based Handover Prediction for LTE Systems. In *2009 International Symposium on Computer Network and Multimedia Technology*, pages 1–4, Jan 2009.

[GWZ16] Jeff Goldberg, Magnus Westerlund, and Thomas Zeng. A Network Address Translator (NAT) Traversal Mechanism for Media Controlled by the Real-Time Streaming Protocol (RTSP). RFC 7825, December 2016.

[HBC+16] S. Hu, W. Bai, K. Chen, C. Tian, Y. Zhang, and H. Wu. Providing bandwidth guarantees, work conservation and low latency simultaneously in the cloud. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[HG12] Rémi Houdaille and Stéphane Gouache. Shaping HTTP Adaptive Streams for a Better User Experience. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 1–9. ACM, 2012.

[HHH+12] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 225–238, New York, NY, USA, 2012. ACM.

[HJM+14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198, New York, NY, USA, 2014. ACM.

[HJMT+16] Toke Hoeiland-Joergensen, Paul McKenney, Dave Taht, Jim Gettys, and Eric Dumazet. The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm. Internet-Draft draft-ietf-aqm-fq-codel-06, IETF Secretariat, March 2016.

[HL99] Hyoung-Kee Choi and J. O. Limb. A behavioral model of Web traffic. In *Proceedings. Seventh International Conference on Network Protocols*, pages 327–334, Oct 1999.

[HMW15] Gongbing Hong, James Martin, and James M. Westall. On fairness and application performance of active queue management in broadband cable networks. *Computer Networks*, 91:390 – 406, 2015.

[HQG+13] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *SIGCOMM Comput. Commun. Rev.*, 43(4):363–374, August 2013.

[HRX08a] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 42(5), July 2008.

[HRX08b] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[HSKHV17] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela. Definition of QoE Fairness in Shared Systems. *IEEE Communications Letters*, 21(1):184–187, Jan 2017.

[HSKHV18] Tobias Hoßfeld, Lea Skorin-Kapov, Poul E. Heegaard, and Martín Varela. A new QoE fairness index for QoE management. *Quality and User Experience*, 3(1):4, Feb 2018.

[HSSZ14] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner. Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 111–116, Sept 2014.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Data Mining, Interference, and Prediction*. Springer, 2009.

[HZM14] J. Hao, R. Zimmermann, and H. Ma. GTube: Geo-predictive Video Streaming over HTTP in Mobile Environments. In *Proceedings of the tth ACM Multimedia Systems Conference*, MMSys'14. ACM, 2014.

[Inc] Google Inc. `https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index`.

[IT14] ITU-T. Estimating end-to-end performance in IP networks for data applications. `https://www.itu.int/rec/T-REC-G.1030/en`, 2014.

[ITU10] ITU-T Rec. H.264. Advanced video coding for generic audiovisual services, 2010.

[Jas16] Jason J. Quinlan, Ahmed H. Zahran, Cormac J. Sreenan. Datasets for AVC (H.264) and HEVC (H.265) Evaluation of Dynamic Adaptive Streaming over HTTP (DASH). In *MMSys '16 Proceedings of the 7th ACM Multimedia Systems Conference*, May 2016.

[JCH98] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *eprint arXiv:cs/9809099*, September 1998.

[JD03] Manish Jain and Constantinos Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Trans. Networking*, 11(4), August 2003.

[JD05] M. Jain and C. Dovrolis. End-to-end Estimation of the Available Bandwidth Variation Range. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, pages 265–276, New York, NY, USA, 2005. ACM.

[JSZ14] 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*, 22(1), Feb 2014.

[JTM15] P. Juluri, V. Tamarapalli, and D. Medhi. SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1765–1770, June 2015.

[KAB17] J. Kua, G. Armitage, and P. Branch. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys Tutorials*, 19(3), thirdquarter 2017.

[Kel] Ariane Keller. *tc Packet Filtering and netem*. ETH Zurich.

[KLC+16] A. M. Kakhki, F. Li, D. Choffnes, E. Katz-Bassett, and A. Mislove. BingeOn Under the Microscope: Understanding T-Mobiles Zero-Rating Implementation. In *Internet-QoE*. ACM, 2016.

[KLW01] Charles Krasic, Kang Li, and Jonathan Walpole. The Case for Streaming Multimedia with TCP. In Doug Shepherd, Joe Finney, Laurent Mathy, and Nicholas Race, editors, *Interactive Distributed Multimedia Systems*, pages 213–218, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[KMC17] Jan Willem Kleinrouweler, Britta Meixner, and Pablo Cesar. Improving Video Quality in Crowded Networks Using a DANE. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, pages 73–78, New York, NY, USA, 2017. ACM.

[LDJ+15] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis. CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks. In *HotMobile*. ACM, 2015.

[LDU+15] 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*, 61(4):651–665, Dec 2015.

[LFCZ11] W. Luo, X. Fang, M. Cheng, and X. Zhou. An optimized handover trigger scheme in LTE systems for high-speed railway. In *Proceedings of the Fifth International Workshop on Signal Design and Its Applications in Communications*, pages 193–196, Oct 2011.

[LFTP+14] J. Le Feuvre, J-M. Thiesse, M. Parmentier, M. Raulet, and C. Daguet. Ultra High Definition HEVC DASH Data Set. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, 2014.

[LGC07] Jeongeun Julie Lee, Maruti Gupta, and Intel Corp. A new traffic model for current user web browsing behavior. 2007.

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6. ACM, 2010.

[LMS04] L. Lenzini, E. Mingozzi, and G. Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *IEEE/ACM Transactions on Networking*, 12(4):681–693, Aug 2004.

[LMT12] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94, New York, 2012.

[LMT+13] Stefan Lederer, Christopher Mueller, Christian Timmerer, Cyril Concolato, Jean Le Feuvre, and Karel Fliegel. Distributed DASH Dataset. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 131–135, New York, NY, USA, 2013. ACM.

[LPY+16] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang. Mobileinsight: Extracting and Analyzing Cellular Network Information on Smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, pages 202–215, New York, NY, USA, 2016. ACM.

[LRW+17] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 183–196, New York, NY, USA, 2017. ACM.

[LWD10] Chao Liu, Ryen W. White, and Susan Dumais. Understanding Web Browsing Behaviors Through Weibull Analysis of Dwell Time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 379–386. ACM, 2010.

[LXW+15] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi. A measurement study on TCP behaviors in HSPA+ networks on

high-speed rails. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2731–2739, April 2015.

[LZG+14] 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*, 32(4):719–733, April 2014.

[Mer14] Peter R. Mercer. *Famous Inequalities*, pages 25–52. Springer New York, New York, NY, 2014.

[MFA15] A. Mansy, M. Fayed, and M. Ammar. Network-layer fairness for adaptive video streams. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, May 2015.

[MGT00] Vishal Misra, Wei-Bo Gong, and Don Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 151–160, New York, NY, USA, 2000. ACM.

[MHC+19] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifa Han, Feng Li, and Jin Li. Realtime Mobile Bandwidth Prediction Using LSTM Neural Network. In David Choffnes and Marinho Barcellos, editors, *Passive and Active Measurement*, pages 34–47, Cham, 2019. Springer International Publishing.

[MHT10] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *J. Mach. Learn. Res.*, 11:2287–2322, August 2010.

[MNA17] H. Mao, R. Netravali, and M. Alizadeh. Neural Adaptive Video Streaming with Pensieve. In *SIGCOMM*. ACM, 2017.

[MSBZ10] M. Mirza, J. Sommers, P. Barford, and X. Zhu. A Machine Learning Approach to TCP Throughput Prediction. *IEEE/ACM Transactions on Networking*, 2010.

[MSMO97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82, July 1997.

[MTAA+16] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi. Video Through a Crystal Ball: Effect of Bandwidth Prediction Quality on Adaptive Streaming in Mobile Environments. In *MoVid*. ACM, 2016.

[MTRAR+12] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521 – 530, 2012.

[MVSA13] Ahmed Mansy, Bill Ver Steeg, and Mostafa Ammar. SABRE: A Client Based Technique for Mitigating the Buffer Bloat Effect of Adaptive Video Flows. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 214–225. ACM, 2013.

[MZA+18] T. Mangla, E. Zegura, M. Ammar, E. Halepovic, K.-W. Hwang, R. Jana, and M. Platania. VideoNOC: Assessing Video QoE for Network Operators Using Passive Measurements. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys'18. ACM, 2018.

[NJ12] Kathleen Nichols and Van Jacobson. Controlling Queue Delay. *Queue*, 10(5):20:20–20:34, May 2012.

[PDT07] Ravi S. Prasad, Constantine Dovrolis, and Marina Thottan. Router Buffer Sizing Revisited: The Role of the Output/Input Capacity Ratio. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07, pages 15:1–15:12, New York, NY, USA, 2007. ACM.

[PFC+15] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, October 2015.

[PJK+18] Michele Polese, Rittwik Jana, Velin Kounev, Ke Zhang, Supratim Deb, and Michele Zorzi. Machine Learning at the Edge: A Data-Driven Architecture with Applications to 5G Cellular Networks. *arXiv e-prints*, page arXiv:1808.07647, Aug 2018.

[PMT12] R. Pries, Z. Magyari, and P. Tran-Gia. An HTTP web traffic model based on the top one million visited web pages. In *Proceedings of*

the 8th Euro-NF Conference on Next Generation Internet NGI 2012, pages 133–139, June 2012.

[PNP+13] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. PIE: A lightweight control scheme to address the bufferbloat problem. In *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, pages 148–155, July 2013.

[PT19] S. D. Patil and M. P. Tahiliani. Towards a better understanding and analysis of controlled delay (CoDel) algorithm by using fluid modelling. *IET Networks*, 8(1):59–66, 2019.

[QHP+18] Yanyuan Qin, Shuai Hao, K. R. Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. Abr streaming of vbr-encoded videos: Characterization, challenges, and solutions. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, pages 366–378, New York, NY, USA, 2018. ACM.

[QJHG16] 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*, ATC '16, pages 1–6, New York, NY, USA, 2016. ACM.

[QS18] Jason J. Quinlan and Cormac J. Sreenan. Multi-profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 375–380, New York, NY, USA, 2018. ACM.

[QZRS15] J. J. Quinlan, A. H. Zahran, K. K. Ramakrishnan, and C. J. Sreenan. Delivery of adaptive bit rate video: balancing fairness, efficiency and quality. In *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, pages 1–6, April 2015.

[QZS16] J. J. Quinlan, A. H. Zahran, and C. J. Sreenan. Datasets for AVC (H.264) and HEVC (H.265) Evaluation of Dynamic Adaptive Streaming over HTTP (DASH). In *MMSys*. ACM, 2016.

[RGR+17] A. Raake, M. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *2017 Ninth*

International Conference on Quality of Multimedia Experience (QoMEX), pages 1–6, May 2017.

[RVGH13] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 114–118. ACM, 2013.

[RZS+17] Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan. Back to the future: Throughput prediction for cellular networks using radio kpis. In *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, HotWireless '17, pages 37–41, New York, NY, USA, 2017. ACM.

[RZS+18a] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello. Incorporating Prediction into Adaptive Streaming Algorithms: A QoE Perspective. In *Proceedings of the 28th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'18. ACM, 2018.

[RZS18b] Aleksandr Reviakin, Ahmed H. Zahran, and Cormac J. Sreenan. Dashc: A Highly Scalable Client Emulator for DASH Video. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 409–414, New York, NY, USA, 2018. ACM.

[SBB+18] Alassane Samba, Yann Busnel, Alberto Blanc, Philippe Dooze, and Gwendal Simon. Predicting file downloading time in cellular network: Large-Scale analysis of machine learning approaches. *Computer Networks*, 145:243 – 254, 2018.

[SBS16] D. Stynes, K. N. Brown, and C. J. Sreenan. A probabilistic approach to user mobility prediction for wireless services. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 120–125, Sept 2016.

[SDV+17] Ahmed Saeed, Nandita Dukkipati, Vytautas Valancius, Vinh The Lam, Carlo Contavalli, and Amin Vahdat. Carousel: Scalable Traffic Shaping at End Hosts. In *Proceedings of the Confer-*

ence of the ACM Special Interest Group on Data Communication, SIGCOMM '17, pages 404–417. ACM, 2017.

[SES+15] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *Communications Surveys Tutorials, IEEE*, 17(1):469–492, Firstquarter 2015.

[SGFS15] Z. Sayeed, E. Grinshpun, D. Faucher, and S. Sharma. Long-term application-level wireless link quality prediction. In *2015 36th IEEE Sarnoff Symposium*, Sept 2015.

[SME15] Y. Sani, A. Mauthe, and C. Edwards. Modelling Video Rate Evolution in Adaptive Bitrate Selection. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 89–94, Dec 2015.

[SME17] Y. Sani, A. Mauthe, and C. Edwards. Adaptive Bitrate Selection: A Survey. *IEEE Communications Surveys Tutorials*, 19(4), 2017.

[SOHW12] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans. Cir. and Sys. for Video Technol.*, 22(12):1649–1668, December 2012.

[SSS18] K. Spiteri, R. Sitaraman, and D. Sparacio. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys'18, pages 123–137, 2018.

[Sto11] Thomas Stockhammer. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 133–144, New York, NY, USA, 2011. ACM.

[SUS16] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[SV96] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, Jun 1996.

[SV98] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, Oct 1998.

[SVL14] I. Sutskever, O. Vinyals, and Q. V Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[SYJ+16] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 272–285, New York, NY, USA, 2016. ACM.

[TKV18] D. Tsilimantos, T. Karagkioules, and S. Valentin. Classifying Flows and Buffer State for Youtube's HTTP Adaptive Streaming Service in Mobile Networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys'18. ACM, 2018.

[TZS14] I. Tsompanidis, A. H. Zahran, and C. J. Sreenan. Mobile network traffic: A user behaviour model. In *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, pages 1–8, May 2014.

[VBNP16] Matteo Varvello, Jeremy Blackburn, David Naylor, and Konstantina Papagiannaki. EYEORG: A Platform For Crowdsourcing Web Quality Of Experience Measurements. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 399–412. ACM, 2016.

[vdHPW+16] 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*, 20(11):2177–2180, 2016.

[vdHPW+18] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Tom Bostoen, and Filip De Turck. An HTTP/2 Push-Based Approach for Low-Latency Live Streaming with Super-

Short Segments. *Journal of Network and Systems Management*, 26(1):51–78, Jan 2018.

[VKD02] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A Mechanism for Background Transfers. *SIGOPS Oper. Syst. Rev.*, 36(SI):329–343, December 2002.

[VS94] Curtis Villamizar and Cheng Song. High Performance TCP in ANSNET. *SIGCOMM Comput. Commun. Rev.*, 24(5):45–60, October 1994.

[WB13] K. Winstein and H. Balakrishnan. TCP Ex Machina: Computer-generated Congestion Control. *SIGCOMM Comput. Commun. Rev.*, 43(4):123–134, August 2013.

[WDM01] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network*, 15(3):28–37, May 2001.

[Wil16] Granville Tunnicliffe Wilson. Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1. *Journal of Time Series Analysis*, 37(5):709–711, 2016. 10.1111/jtsa.12194.

[WKHC14] H-L. Wang, S-J. Kao, C-Y. Hsiao, and F-M. Chang. A moving direction prediction-assisted handover scheme in LTE networks. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):190, Nov 2014.

[WS14] Sheng Wei and Viswanathan Swaminathan. Low Latency Live Video Streaming over HTTP 2.0. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, pages 37:37–37:42, New York, NY, USA, 2014. ACM.

[WSB13] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 459–472, Berkeley, CA, USA, 2013. USENIX Association.

[WSBL03] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[XHY+05] Nai-xue Xiong, Yan-xiang He, Yan Yang, Bin Xiao, and Xiaohua Jia. On Designing a Novel PI Controller for AQM Routers Supporting TCP Flows. In Yanchun Zhang, Katsumi Tanaka, Jeffrey Xu Yu, Shan Wang, and Minglu Li, editors, *Web Technologies Research and Development - APWeb 2005*, pages 991–1002, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[XMML13] Qiang Xu, Sanjeev Mehrotra, Zhuoqing Mao, and Jin Li. PROTEUS: Network Performance Forecast for Real-time, Interactive Mobile Applications. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 347–360, New York, NY, USA, 2013. ACM.

[XXW+14] Q. Xiao, K. Xu, D. Wang, L. Li, and Y. Zhong. TCP Performance over Mobile Networks in High-Speed Mobility Scenarios. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 281–286, Oct 2014.

[XZKL15] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. piStream: Physical Layer Informed Adaptive Video Streaming over LTE. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 413–425. ACM, 2015.

[XZKL16] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. piStream: Physical Layer Informed Adaptive Video Streaming Over LTE. *GetMobile: Mobile Comp. and Comm.*, 20(2):31–34, October 2016.

[YAZ+19] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Continual learning improves Internet video streaming. *arXiv e-prints*, page arXiv:1906.01113, Jun 2019.

[YJS+18] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei. Linkforecast: Cellular link bandwidth prediction in lte networks. *IEEE*

Transactions on Mobile Computing, 17(7):1582–1594, July 2018.

[YJSS15a] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.*, 45(4), August 2015.

[YJSS15b] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 325–338. ACM, 2015.

[YTHL19] S. Yang, Y. Tseng, C. Huang, and W. Lin. Multi-Access Edge Computing Enhanced Video Streaming: Proof-of-Concept Implementation and Prediction/QoE Models. *IEEE Transactions on Vehicular Technology*, 68(2):1888–1902, Feb 2019.

[YXC17] Z. Yan, J. Xue, and C. W. Chen. Prius: Hybrid Edge Cloud and Client Adaptation for HTTP Adaptive Streaming in Cellular Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):209–222, Jan 2017.

[ZEG+15] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. Can Accurate Predictions Improve Video Streaming in Cellular Networks? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, pages 57–62, New York, NY, USA, 2015. ACM.

[ZFT18] Anatoliy Zabrovskiy, Christian Feldmann, and Christian Timmerer. Multi-codec DASH Dataset. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 438–443, New York, NY, USA, 2018. ACM.

[ZJB+14] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer. Dynamic application-aware resource management using Software-Defined Networking: Implementation prospects and challenges. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–6, May 2014.

[ZPC+15] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive Congestion Control for

Unpredictable Cellular Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 509–522, New York, NY, USA, 2015. ACM.

[ZPH19] C. Zhang, P. Patras, and H. Haddadi. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Communications Surveys Tutorials*, 21(3):2224–2287, thirdquarter 2019.

[ZQR+16] Ahmed H. Zahran, Jason Quinlan, Darijo Raca, Cormac J. Sreenan, Emir Halepovic, Rakesh K. Sinha, Rittwik Jana, and Vijay Gopalakrishnan. Oscar: An optimized stall-cautious adaptive bitrate streaming algorithm for mobile networks. In *Proceedings of the 8th International Workshop on Mobile Video*, MoVid '16, pages 2:1–2:6, New York, NY, USA, 2016. ACM.

[ZQRS17] Ahmed H. Zahran, Jason J. Quinlan, K. K. Ramakrishnan, and Cormac J. Sreenan. SAP: Stall-Aware Pacing for Improved DASH Video Experience in Cellular Networks. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, pages 13–26. ACM, 2017.

[ZRS18] A. H. Zahran, D. Raca, and C. Sreenan. ARBITER+: Adaptive Rate-Based InTElligent HTTP StReaming Algorithm for Mobile Networks. *IEEE Transactions on Mobile Computing*, 2018.

Acronyms

ACF Autocorrelation Function. 43, 44

ADB Android Debug Bridge. 45, 64

AI Artificial Intelligence. 4

AIMD Adaptive Increase Multiplicative Decrease. 16

AP Access Point. 45, 46

API Application Programming Interface. 43, 78, 84, 106, 109

AQM Active Queue Management. xiii, 6, 7, 22, 116–121, 129, 131, 134, 136, 139, 141–145, 164, 167, 170

ARIMA Autoregressive Integrated Moving Average. 18

AV1 AOMedia Video 1. 14, 24

AVC Advanced Video Coding. 14, 27, 48

BDP Bandwidth Delay Product. 21, 117–119, 122–124, 126–128, 130, 131, 134, 136, 138, 141–143, 151, 158, 160, 167

BLER Block Error Rate. 36

BS base station. 2, 5, 18, 75

CBR Constant Bit-Rate. 24

CDF Cumulative Distribution Function. 97, 155, 156

CDN Content Delivery Networks. 15, 23, 135, 141

CoD Coefficient of Determination. 87

CoDel Controlled Delay Management. xiii, 119, 121, 129, 131, 134, 135, 139, 141, 142, 144, 145, 148, 149, 151, 155, 156, 158, 160, 162, 164, 167

CoV Coefficient of Variation. 94

CQI Channel Quality Indicator. 17, 36, 40, 41, 83, 96, 111

CV Cross-Validation. 85, 87, 88

DASH Dynamic Adaptive Streaming over HTTP. 23, 24, 46

DL Deep Learning. 4

DRR Deficit Round Robin. 148, 152

DTX Discontinuous Transmission Ratio. 17

ECN Explicit Congestion Notification. 22

eNodeB evolved Node B. 2, 3, 19, 26, 32, 33, 36, 40–43, 108

EWMA Exponential Weighted Moving Average. 3, 32, 50, 61–63, 66, 67, 78, 88, 99, 100, 152

FIFO First In First Out. 21, 22, 107, 118, 119, 129, 131, 142, 143, 145, 148, 149, 151, 158, 160, 162

FQ Flow Queue. 119, 129, 131, 134, 135, 139, 141, 142, 144, 156

GAN Generative Adversarial Network. 33

GPS Global Positioning System. 3, 25, 26, 31–33, 35, 36, 41–43

HAS HTTP adaptive streaming. xii, xiii, 1–3, 5–8, 10–19, 21, 23–33, 44, 46, 47, 49, 51, 54, 57, 59, 60, 62–64, 67–71, 74–76, 78, 82, 100, 101, 103, 107, 115–118, 120, 121, 129, 130, 141–146, 165–168, 171

HD High Definition. 24, 27

HEVC High Efficiency Video Coding. 14, 48

HLS HTTP Live Streaming. 46

HTTP HyperText Transfer Protocol. 1, 10, 12–15, 18, 21, 29, 31, 47, 53, 118, 142, 145, 164

HTTPS Hypertext Transfer Protocol Secure. 162

IP Internet Protocol. 10, 11

LR Latency Rate. 148

LSTM Long Short-Term Memory. 17, 113, 114

LTE Long-Term Evolution. 18, 111

MCS Modulation and Coding Scheme. 18, 36

ML Machine Learning. 4–6, 18, 75, 76, 83–85, 94, 97, 106–108, 113, 115, 166, 168, 171

ML/DL Machine and Deep Learning. 4, 5

MPC Model Predictive Control. 16

MTU Maximum Transmission Unit. 148, 154, 162

NAT Network Address Translation. 12

OS Operating System. 19, 23, 45, 77

PID Proportional Integral Derivative. 170

PLT Page Loading Time. xiii, 27, 28, 57, 118, 134, 143, 148, 149, 152, 160, 162, 164, 167

PQ Priority Queuing. 22

PRB Physical resource Block. 18, 19, 111

QoE Quality of Experience. vi, xii, xiii, 2, 6–8, 13, 15, 16, 18, 23, 27–31, 49, 51, 54–57, 59, 60, 62, 64–66, 68, 71, 74–76, 78, 101–106, 108, 109, 115, 117, 119, 121–131, 134–136, 138, 139, 142–144, 146, 147, 149, 151, 152, 157, 158, 160, 162, 164, 166–168, 171

QoS Quality of Service. 11, 27, 56

QUIC Quick UDP Internet Connections. 12

RAN Radio Access Network. 84

RE Resource Element. 35

RED Random Early Detection. 21, 121

RF Random Forest. 83–85, 87, 113

RLS Recursive Least Squares. 17

RNN Recurrent Neural Networks. 113

RSRP Reference Signal Received Power. 35, 36, 82, 96, 97, 111

RSRQ Reference Signal Received Quality. 35, 36, 82, 96, 111

RSSI Received Signal Strength Indicator. 35, 43

RTCP Real-Time Control Protocol. 11

RTMP Real-Time Messaging Protocol. 11

RTP Real-Time Transport Protocol. 11, 13, 15

RTSP Real-Time Streaming Protocol. 11, 12

RTT round-trip-time. xiii, 47, 117–120, 122, 125–127, 135, 136, 138, 139, 141–143, 151, 167

SDN Software-Defined Networking. 22

SINR Signal to Interference and Noise Ratio. 18, 36

SNI Server Name Identification. 162

SNR Signal to Noise Ratio. 35, 83, 96, 111

SoC System on Chip. 33, 34, 108

SVM Support Vector Machines. 113, 114

TCP Transmission Control Protocol. 3, 6, 12, 15–17, 20–22, 26, 31, 34, 47, 52, 53, 116, 117, 127, 128, 136, 139, 144, 146, 170

UDP User Datagram Protocol. 12, 19

UE User Equipment. 26, 36

UPPLT User Perceived Page Loading Time. 27, 28

VBR Variable Bit-Rate. 21, 24, 25, 116

VoD Video on Demand. 12, 14

VR Virtual Reality. 32, 33

WFQ Weighted Fair Queuing. 22