2019-12-18

A QoE Fairness Approach for Adaptive Video Streaming

Fisher, Daniel Gregg

UNIVERSITY OF CALGARY

A QoE Fairness Approach for Adaptive Video Streaming

by

Daniel Gregg Fisher

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

DECEMBER, 2019

Abstract

According to Cisco, video traffic accounts for a mjority of all IP traffic globally and the number and type of internet connected devices is growing [8]. DASH (Dynamic Adaptive Streaming over HTTP) has emerged as an effective way to provide these heterogeneous clients a high quality of experience (QoE) when streaming video. In this thesis, we propose FairQ, a client-server DASH adaptation algorithm based on a systematic analysis of DASH techniques and algorithms. FairQ utilizes client's request qualities, buffer levels, resolutions, and SSIM values to not only provide QoE to individual streaming sessions but to assure fairness across streaming sessions. We compared FairQ to a variety of previously proposed DASH algorithms under a variety of network scenarios including a real bandwidth trace. We found that FairQ was able to achieve better fairness than comparable algorithms with an average increase of 53% while producing comparable buffer levels and quality switches under heterogeneous client scenarios.

Acknowledgements

I would like to thank my supervisor, Dr. Mea Wang, for her patience and guidance throughout the long process of this thesis. I am very thankful that she remained calm and supportive when I lost motivation and felt like I had reached a dead end. Without her this thesis wouldn't have been possible. I would also like to thank Cyriac James for his help with technical issues throughout this thesis and for being a positive example of someone with endless persistence and drive. I would also like to thank my parents for supporting me throughout this process.Finally, I would like to thank the University of Calgary for putting up with me for so many years.

Table of Contents

Abstract . . . ii
Acknowledgements . . . iii
Table of Contents . . . iv
List of Tables . . . vi
List of Figures . . . x
List of Symbols . . . xiii
1 Introduction . . . 1
2 Background and Related Work . . . 8
2.1 Overview of DASH Adaptation Algorithms . . . 10
2.1.1 Rate-based Adaptation (client side) . . . 10
2.1.2 Buffer-based Adaptation (client or server side) . . . 11
2.1.3 Hybrid of Rate-based and Buffer-based Adaptation (client or server side) . . . 12
2.2 Fairness Issues across Multiple DASH Sessions . . . 16
2.2.1 Client-side solutions . . . 16
2.2.2 Network Solutions . . . 19
2.2.3 QoE Fairness Algorithms . . . 20
3 Analysis of QoE and Fairness in DASH . . . 24
3.1 Experimental Setup . . . 27
3.1.1 Setup . . . 28
3.1.2 Validating our Experimental Setup . . . 32
3.2 Overhead of DASH . . . 34
3.3 Comparison of Adaptive Approaches . . . 38
3.4 Harmonic Mean Window Tests . . . 43
3.5 Initial Phase . . . 47
3.6 Fairness Across Sessions . . . 53
3.7 Summary . . . 59
4 Comparison of Adaptation Algorithms . . . 61
4.1 Experimental Setup . . . 61
4.2 Bandwidth Drops . . . 64
4.2.1 Homogeneous Client Scenarios . . . 65
4.2.2 Mixed Client Scenarios . . . 68
4.2.3 Examining the 1080p Clients Under Mixed Client Scenarios . . . 71
4.2.4 Examining the 360p Clients Under Mixed Client Scenarios . . . 73
4.3 Bandwidth Drops and Raises . . . 75
4.3.1 Homogeneous Client Scenarios . . . 75
4.3.2 Mixed Client Scenarios . . . 78
4.3.3 Examining the 1080p Clients Under Mixed Client Scenarios . . . 79
4.4 Real Bandwidth Trace . . . 80
4.4.1 Mixed Client Scenario . . . 81
4.4.2 Examining the 1080p Clients Under Mixed Client Scenarios . . . 84
4.4.3 Examining the 360p Clients Under Mixed Client Scenarios . . . 85
4.5 Summary . . . 88

5 FairQ Adaptation Scheme . . . 90
5.1 Design Objectives . . . 90
5.2 Design Overview of FairQ . . . 93
5.2.1 Quality Smoothing . . . 98
5.2.2 Fairness . . . 99
5.3 Design Objectives Revisited . . . 100
5.4 Complexity Analysis . . . 102
5.5 Implementation . . . 103
5.5.1 Parameters . . . 105
5.5.2 Asynchronous Issues . . . 105
5.6 Summary . . . 108
6 Evaluation . . . 110
6.1 Experimental Setup . . . 110
6.2 Default Settings . . . 111
6.3 Fairness . . . 114
6.4 Utilization . . . 119
6.5 Quality . . . 120
6.6 Switches . . . 122
6.7 Buffer . . . 123
6.8 Initial Phase Length . . . 126
6.9 Overhead of FairQ . . . 127
6.10 Summary . . . 128
7 Conclusion and Future Work . . . 130
7.1 Thesis Summary . . . 130
7.2 Future Work . . . 133
Bibliography . . . 137
A Overall Results from Chapter 4 and 6 . . . 145
A.1 Bandwidth Drops . . . 145
A.1.1 Homogeneous Client Scenario . . . 145
A.1.2 Mixed Client Scenario . . . 147
A.1.3 Homogeneous Within Mixed Client Scenario . . . 149
A.2 Bandwidth Drops and Raises . . . 153
A.2.1 Homogeneous Client Scenarios . . . 153
A.2.2 Mixed Client Scenarios . . . 155
A.2.3 Homogeneous Within Mixed Client Scenarios . . . 156
A.3 Real Bandwidth Trace . . . 159
A.3.1 Mixed Client Scenarios . . . 159
A.3.2 Homogeneous Within Mixed Client Scenario . . . 160
B Comparison of FairQ Variations by Metric . . . 161
B.1 Fairness Tables . . . 161
B.2 Utilization Tables . . . 163
B.3 Quality Tables . . . 165
B.4 Quality Switches Tables . . . 167
B.5 Buffer Tables . . . 169
B.6 Initial Phase Length Tables . . . 171

List of Tables

3.1 Average standard deviation of segment sizes for each video at a segment size of 2 seconds. . . . 30
3.2 Quality Level Information for the "Big Buck Bunny" video at a segment length of 2 seconds from the dataset in [28] . . . 30
3.3 Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the static network scenario. . . . 39
3.4 Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the drop network scenario. . . . 40
3.5 Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the short drop network scenario. . . . 41
3.6 Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the static network scenario. . . . 43
3.7 Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the drop network scenario. . . . 44
3.8 Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the short drop network scenario. . . . 46
3.9 SSIM values for the "Big Buck Bunny" video at 3 resolutions (360p, 720p, 1080p). 56

4.1 Results from the Drop 1 network scenario under a homogeneous client scenario. . . 66
4.2 Results from the Drop 1 network scenario under a mixed client scenario. . . . 70
4.3 Results from the Drop 1 network scenario under a mixed client scenario showing only the 1080p clients. . . . 72
4.4 Results from the Big Drop network scenario under a homogeneous client scenario. 77
4.5 Results from the Big Drop network scenario under a mixed client scenario. . . . 79
4.6 Results from the Big Raise network scenario under a mixed client scenario showing only the 1080p clients. . . . 80
4.7 Results from the Real Bandwidth Trace network scenario under a mixed client scenario. . . . 82
4.8 Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 1080p clients. . . . 84
4.9 Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 360p clients. . . . 86
4.10 The pros and cons of each algorithm over all the tests done in this chapter. . . . 89

6.1 Default settings used during this Chapter that include network scenarios, client scenarios, number of clients, and FairQ parameters. . . . 112
6.2 Results from the Drop 1 network scenario under a homogeneous client scenario. . . 115
6.3 Results from the Drop 1 network scenario under a mixed client scenario. . . . 115
6.4 Fairness results from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. 117
6.5 SSIM values for the "Big Buck Bunny" video at 3 resolutions (360p, 720p, 1080p). 120
6.6 Utilization (%) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. 121

6.7 Quality levels (Mbps) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. . . . 121
6.8 Number of quality switches from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. . . . 124
6.9 Buffer level (seconds) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. . . . 126
6.10 Initial phase length (seconds) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric. . . . 128

A.1 Results from the Drop 1 network scenario under a homogeneous client scenario. . . 145
A.2 Results from the Drop 2 network scenario under a homogeneous client scenario. . . 146
A.3 Results from the Drop 3 network scenario under a homogeneous client scenario. . . 146
A.4 Results from the Drop 4 network scenario under a homogeneous client scenario. . . 147
A.5 Results from the Drop 1 network scenario under a mixed client scenario. 147
A.6 Results from the Drop 2 network scenario under a mixed client scenario. 148
A.7 Results from the Drop 3 network scenario under a mixed client scenario. 148
A.8 Results from the Drop 4 network scenario under a mixed client scenario. 149
A.9 Results from the Drop 1 network scenario under a mixed client scenario showing only the 1080p clients. . . . 149
A.10 Results from the Drop 1 network scenario under a mixed client scenario showing only the 360p clients. . . . 150
A.11 Results from the Drop 2 network scenario under a mixed client scenario showing only the 1080p clients. . . . 150
A.12 Results from the Drop 2 network scenario under a mixed client scenario showing only the 360p clients. . . . 151
A.13 Results from the Drop 3 network scenario under a mixed client scenario showing only the 1080p clients. . . . 151
A.14 Results from the Drop 3 network scenario under a mixed client scenario showing only the 360p clients. . . . 152
A.15 Results from the Drop 4 network scenario under a mixed client scenario showing only the 1080p clients. . . . 152
A.16 Results from the Drop 4 network scenario under a mixed client scenario showing only the 360p clients. . . . 153
A.17 Results from the Big Drop network scenario under a homogeneous client scenario. 153
A.18 Results from the Big Raise network scenario under a homogeneous client scenario. 154
A.19 Results from the Drop and Raise network scenario under a homogeneous client scenario. . . . 154
A.20 Results from the Big Drop network scenario under a mixed client scenario. 155
A.21 Results from the Big Raise network scenario under a mixed client scenario. 155
A.22 Results from the Drop and Raise network scenario under a mixed client scenario. . 156

A.23 Results from the Big Drop network scenario under a mixed client scenario showing only the 1080p clients. . . . 156
A.24 Results from the Big Drop network scenario under a mixed client scenario showing only the 360p clients. . . . 157
A.25 Results from the Big Raise network scenario under a mixed client scenario showing only the 1080p clients. . . . 157
A.26 Results from the Big Raise network scenario under a mixed client scenario showing only the 360p clients. . . . 158
A.27 Results from the Drop and Raise network scenario under a mixed client scenario showing only the 1080p clients. . . . 158
A.28 Results from the Drop and Raise network scenario under a mixed client scenario showing only the 360p clients. . . . 159
A.29 Results from the Real Bandwidth Trace network scenario under a mixed client scenario. . . . 159
A.30 Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 1080p clients. . . . 160
A.31 Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 360p clients. . . . 160

B.1 Fairness produced by FairQ-smooth 15s across network and client scenarios. . . . 161
B.2 Fairness produced by FairQ 15s across network and client scenarios. . . . 161
B.3 Fairness produced by FairQ-smooth 20s across network and client scenarios. . . . 162
B.4 Fairness produced by FairQ 20s across network and client scenarios. . . . 162
B.5 Fairness produced by FairQ-smooth 25s across network and client scenarios. . . . 162
B.6 Fairness produced by FairQ 25s across network and client scenarios. . . . 163
B.7 Utilization produced by FairQ-smooth 15s across network and client scenarios. . . . 163
B.8 Utilization produced by FairQ 15s across network and client scenarios. . . . 163
B.9 Utilization produced by FairQ-smooth 20s across network and client scenarios. . . . 164
B.10 Utilization produced by FairQ 20s across network and client scenarios. . . . 164
B.11 Utilization produced by FairQ-smooth 25s across network and client scenarios. . . . 164
B.12 Utilization produced by FairQ 25s across network and client scenarios. . . . 165
B.13 Quality levels produced by FairQ-smooth 15s across network and client scenarios. . 165
B.14 Quality levels produced by FairQ 15s across network and client scenarios. . . . 165
B.15 Quality levels produced by FairQ-smooth 20s across network and client scenarios. . 166
B.16 Quality levels produced by FairQ 20s across network and client scenarios. . . . 166
B.17 Quality levels produced by FairQ-smooth 25s across network and client scenarios. . 166
B.18 Quality levels produced by FairQ 25s across network and client scenarios. . . . 167
B.19 Quality switches produced by FairQ-smooth 15s across network and client scenarios. 167
B.20 Quality switches produced by FairQ 15s across network and client scenarios. . . . 167
B.21 Quality switches produced by FairQ-smooth 20s across network and client scenarios. 168
B.22 Quality switches produced by FairQ 20s across network and client scenarios. . . . 168
B.23 Quality switches produced by FairQ-smooth 25s across network and client scenarios. 168
B.24 Quality switches produced by FairQ 25s across network and client scenarios. . . . 169
B.25 Buffer levels produced by FairQ-smooth 15s across network and client scenarios. . 169
B.26 Buffer levels produced by FairQ 15s across network and client scenarios. . . . 169

B.27 Buffer levels produced by FairQ-smooth 20s across network and client scenarios. . 170
B.28 Buffer levels produced by FairQ 20s across network and client scenarios. 170
B.29 Buffer levels produced by FairQ-smooth 25s across network and client scenarios. . 170
B.30 Buffer levels produced by FairQ 25s across network and client scenarios. 171
B.31 Initial phase lengths produced by FairQ-smooth 15s across network and client scenarios. 171
B.32 Initial phase lengths produced by FairQ 15s across network and client scenarios. . . 171
B.33 Initial phase lengths produced by FairQ-smooth 20s across network and client scenarios. 172
B.34 Initial phase lengths produced by FairQ 20s across network and client scenarios. . . 172
B.35 Initial phase lengths produced by FairQ-smooth 25s across network and client scenarios. 172
B.36 Initial phase lengths produced by FairQ 25s across network and client scenarios. . . 173

List of Figures and Illustrations

1.1 A DASH server and a group of heterogeneous clients. 2

2.1 A DASH client dynamically adjusting the video bitrate during a streaming session. 8
2.2 Subset of a MPD file for the video "Big Buck Bunny" from dataset in [28] 9
2.3 Pseudo code for buffer based algorithm presented in [23] 13
2.4 Pseudo code default GPAC algorithm 15
2.5 Unsynchronized segment downloads. 17
2.6 Synchronized segment downloads. 17
2.7 Pseudo code for FESTIVE algorithm presented in [21] 18
2.8 Pseudo code for variation of algorithm presented in [10] 21

3.1 The default network setup where clients connect directly to the video server. 28
3.2 The proxy network setup where clients connect to a proxy placed between them and the video server. 29
3.3 How the Linux tool `tc` controls bandwidth by associating each incoming packet with a token from the token bucket. 31
3.4 Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 2 ms respectively. 33
3.5 Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 20 ms respectively. 33
3.6 Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 200 ms respectively. 34
3.7 Buffer levels when a client requests only segments with a quality level of 222 Kbps and the bandwidth limit set to 222 Kbps. 35
3.8 Listed bitrate in the MPD file vs experimentally found minimum bandwidth of quality levels for the "Big Buck Bunny" video. 36
3.9 Percent difference between listed bitrate in the MPD file and the experimentally found bandwidth for each quality level of the video "Big Buck Bunny". 36
3.10 Buffer levels when a client requests only segments with a quality level of 222 Kbps and the bandwidth limit set to 236 Kbps. 37
3.11 The drop network scenario. 40
3.12 Short drop network scenario. 42
3.13 Quality levels of 4 clients over time using the harmonic mean of the previous 30 throughput values under the drop network scenario. 45
3.14 Quality levels of 4 clients over time using the harmonic mean of the previous 20 throughput values under the drop network scenario. 45
3.15 Quality levels of 4 clients over time using the harmonic mean of the previous 30 throughput values under the short drop network scenario. 46
3.16 Quality levels of 4 clients over time using the harmonic mean of the previous 20 throughput values under the short drop network scenario. 47
3.17 Buffer levels with a harmonic mean window of 20 using the default behaviour. . . . 49
3.18 Buffer levels with a harmonic mean window of 20 using the lowest quality strategy. 49
3.19 Buffer levels with a harmonic mean window of 20 using the midway quality strategy. 50

3.20 Quality levels with a harmonic mean window of 20 using the default behaviour. . . 51
3.21 Quality levels with a harmonic mean window of 20 when using the lowest quality strategy. . . . 51
3.22 Quality levels with a harmonic mean window of 20 when using the midway quality strategy. . . . 52
3.23 Quality levels of 2 clients using the harmonic mean of the previous 20 throughput values with a bandwidth limit of 2.64 Mbps. . . . 53
3.24 Quality levels of 2 clients using the FESTIVE algorithm with a bandwidth limit of 2.64 Mbps. . . . 54
3.25 Quality levels with a bandwidth limit of 2.64 Mbps and a proxy modifying client requests. . . . 55
3.26 Comparison of SSIM values for the "Big Buck Bunny" video at 3 resolutions (360p, 720p, 1080p). . . . 57
3.27 Quality levels of 2 clients using the resolution-based algorithm with a bandwidth limit of 2.64 Mbps. . . . 58

4.1 Download rate of segments when a client connects to the server with no bandwidth limit in place. . . . 63
4.2 Drop 1 to 4 network scenarios. . . . 65
4.3 Quality levels for the hybrid algorithm during the Drop 1 network scenario under the homogeneous client scenario. . . . 68
4.4 Quality levels for FESTIVE during the Drop 1 network scenario under the homogeneous client scenario. . . . 69
4.5 Quality levels for the rate-based algorithm during the Drop 3 network scenario under the homogeneous client scenario. . . . 69
4.6 Quality levels of 360p hybrid clients during the Drop 1 network scenario under the homogeneous client scenario. . . . 74
4.7 Quality levels of 360p buffer-based clients during the Drop 1 network scenario under the homogeneous client scenario. . . . 74
4.8 Big Drop, Big Raise, and Drop and Raise network scenarios. . . . 76
4.9 Quality levels for the buffer-based algorithm under the Drop and Raise network scenario. . . . 78
4.10 Bandwidth under the real bandwidth trace network scenario. . . . 81

5.1 Showing the initial phase, optional transition phase, and adaptation phase. . . . 94
5.2 Showing the first request of *n* clients where each client sends their screen resolution along with the segment quality. . . . 95
5.3 Clients during the initial phase where FairQ groups clients based on their screen resolutions and sends them the midway quality level. . . . 96
5.4 Overview of the FairQ algorithm showing how FairQ loops through resolution groups based on average SSIM values and raises quality levels of high buffer clients. 97
5.5 The optional transition phase that occurs between the initial phase and adaptation phase when quality smoothing is turned on vs no transition phase when quality smoothing is turned off. . . . 98
5.6 SSIM values for the video Big Buck Bunny at 3 resolutions (360p, 720p, 1080p). . 99

5.7 PSNR values for the video Big Buck Bunny at 3 resolutions (360p, 720p, 1080p). . 100
5.8 Pseudo code for FairQ where r_i, q_i, f_{q_i}, and $buff_i$ are the request quality, quality level, fair quality level, and buffer level of client i and $buff_{low}$ is the low buffer threshold. 104
5.9 An overview of the FairQ algorithm showing the modules and the interaction between the proxy, video server, and clients. 106
5.10 Illustrating four cases of phase level asynchronicity. 108

6.1 The network scenarios that will be analyzed during this Chapter. 113
6.2 Quality levels of 1080p FairQ clients and 1080p resolution-based clients under the Big Drop network scenario. 116
6.3 Quality levels of 360p FairQ clients and 360p resolution-based clients under the Drop 1 network scenario. 118
6.4 Buffer levels of a rate-based 1080p client under the Real Bandwidth Trace. 123
6.5 Buffer and Quality levels over time of a 360p and 1080p FairQ client during the Real Bandwidth Trace. 124
6.6 Quality levels of a 360p and 1080p resolution-based client under the Real Bandwidth Trace. 125
6.7 Quality levels during the initial phase of a FairQ client compared to a resolution-based client. 127
6.8 CPU usage of FairQ on the proxy during a run using the real bandwidth trace and 20 clients. 129

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
DASH	Dynamic Adaptive Streaming over HTTP
QoE	Quality of Experience
BW	Bandwidth
`tc`	Traffic Control
Bunny	Big Buck Bunny video
U of C	University of Calgary

Chapter 1

Introduction

Cisco reports suggest that video traffic will account for 82% of all IP traffic globally by 2022, up from 75% in 2017 [8]. According to the 2018 Global Internet Phenomena Report by Sandvine, Netflix alone accounts for 15% of all downstream traffic in the entire internet [48]. Cisco reports also suggest that the number and type of internet connected devices are growing each year, creating more and more heterogeneous network environments [8]. To accommodate such a diverse range of clients, adaptive video streaming techniques have been widely adopted [35, 1, 20]. Adaptive streaming provides clients with a range of quality levels for a single video and allows clients to choose which quality level to stream at. This ensures that each client can receive a video quality that is appropriate for their network and the physical characteristics of their device, such as screen size.

Dynamic Adaptive Streaming over HTTP (DASH) is a HTTP-based adaptive streaming technique that allows a video server to accommodate a wide array of heterogeneous clients from 4K televisions to smartphones as shown in Fig. 1.1. By using HTTP, DASH allows existing HTTP infrastructure, such as caches and web servers, to be seamlessly integrated. DASH is able to accommodate a wide array of heterogeneous devices by providing users with a range of quality levels for each video being offered by a DASH server. As shown in Fig. 1.1, clients can range from a smart phone, to a laptop, to a television. These clients are likely to have different bandwidth requirements and network conditions. By offering several quality levels and allowing clients to choose quality levels, a DASH server is able to provide each of these clients a suitable video quality and an overall high Quality-of-Experience (QoE).

Common objective measures of QoE are visual quality and visual stability. The visual quality is commonly measured via Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index

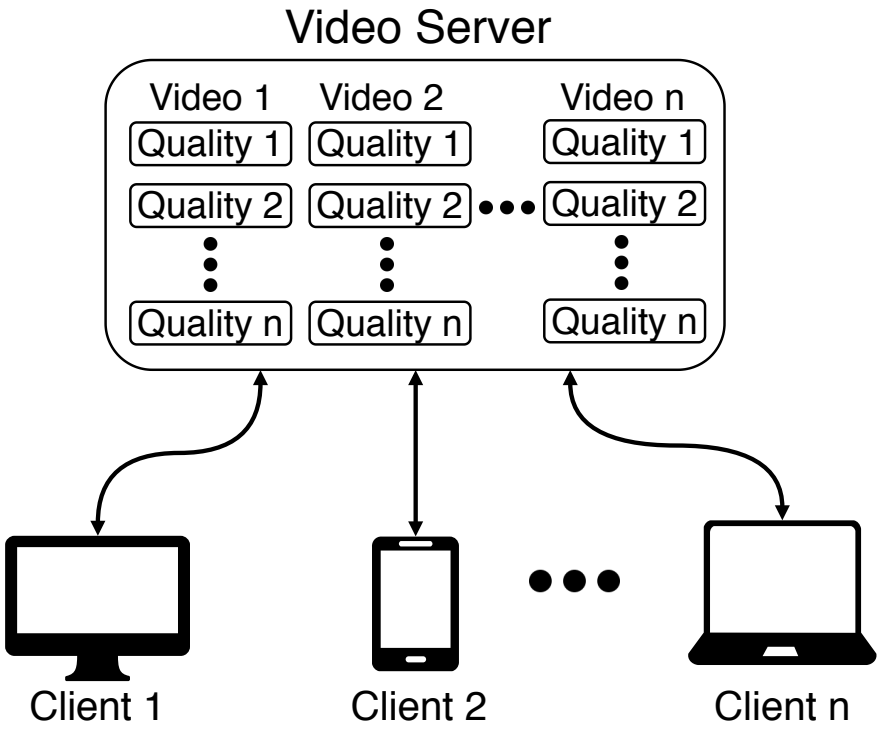

Figure 1.1: A DASH server and a group of heterogeneous clients.

(SSIM) [62]. These two measurements have a direct correlation to the bitrate (in bps) of a video. Improvements to encoding techniques to increase PSNR [44, 15] and SSIM [55, 57] is an active area of research and outside the scope of this thesis. DASH often uses the average bitrate of videos as a measure of visual quality. Videos on DASH servers are ordered from lowest to highest bitrate with the lowest quality level having the lowest bitrate. Visual stability measures how often the video quality changes and how often the video playback stalls due to an empty buffer. It has been shown that frequent changes in video quality and stalls in playback negatively affect QoE [47, 37, 17].

Early streaming protocols, such as Real-time Transport Protocol (RTP) [50], had players use a buffer to download and store video chunks before playback to prevent stalls due to an unstable network or network jitter. Most video players utilize this technique today, including DASH players. The number of video chunks currently in the buffer is referred to as the buffer level and is generally measured in seconds. Higher buffer levels are preferable to lower buffer levels as they allow a

video player to cope with jitter and network changes. RTP and other similar protocols, such as Real Time Messaging Protocol (RTMP) [39], generally provided a single bitrate that was well below the bandwidth of an average viewer and thus did not allow for any adaptation. This achieved visually stability however, higher bandwidth clients would experience low bandwidth utilization. More recent streaming techniques, such as DASH [20], Microsoft Smooth Streaming [35], and Adobe HTTP Dynamic Streaming [1], are adaptive streaming techniques that allow clients to change the visual quality mid stream to better adapt to current network conditions.

As shown in Fig. 1.1, a DASH server prepares multiple quality levels for all the videos being offered. The number of quality levels and the bitrate for each quality level for a video is provided by the DASH server in a Media Presentation Description (MPD) file in XML format. To begin a DASH session, the users video player first downloads the MPD file for the video of interest to determine the bitrate of each quality level being offered. Once a user begins a streaming session, their video player can monitor their current bandwidth and switch to the quality level that best fits their current bandwidth mid stream. This allows video streams to dynamically adapt to a users potentially unreliable or unstable network.

A very simple way to estimate the current bandwidth is to use the download rate, also called the throughput, of the most recently downloaded video chunk. If the currently available bandwidth were to suddenly drop mid stream, the users video player could recognize this and switch to a lower, more appropriate quality level. This would result in a drop in video quality, however this is preferable to staying at the same quality level and experiencing frequent stalls in video playback due to an empty buffer. How a DASH video players estimates currently available bandwidth and how it uses this bandwidth estimation to choose a quality level is referred to as the adaptation algorithm. The MPEG-DASH standard does not define a standard adaptation algorithm and lets researchers and designers of video players create their own [20].

Many researchers have proposed adaptation algorithms that focus on factors within a single streaming session such as buffer levels, segment qualities, and number of quality switches. These

include using the harmonic mean of throughput values [21], the median of throughput values [13], and the weighted average of throughput values based on segment size in bytes [22] as bandwidth estimation. These bandwidth estimations attempt to smooth small fluctuations in network conditions in order to provide a stable estimation that leads to a lower number of quality switches. Huang *et al*. argue that we should do away with attempting to estimate currently available bandwidth and instead focus on observing and controlling the video buffer level [18]. By observing the overall trend of the buffer level, a client can gauge network conditions without having to directly estimate available bandwidth. Kim *et al*. take this a step further by proposing a fully server side, buffer-based adaptation scheme that chooses segment qualities based on estimated client buffer levels [23]. This scheme estimates client buffer levels based on the time between subsequent segment requests and picks quality levels based on the overall trend of this estimated buffer level.

In addition to QoE within a single streaming session, fairness across multiple streaming sessions also plays a vital role in the overall QoE offered by DASH. There are two different approaches to fairness that we will call (1) resource fairness, and (2) QoE fairness. Resource fairness involves giving each client equal network resources. In contrast to this, QoE fairness involves giving clients network resources with the goal of producing equal QoE. QoE is typically quantified using some objective measure such as PSRN or SSIM. Depending on the screen resolution of clients this may or may not produce resource fairness. DASH clients can stream video that is at or below their screen resolution and providing a smartphone and a 4K television with the same bitrate of video is very likely to lead to different QoE. Thus, to achieve QoE fairness the 4K television will have to receive a video at a higher bitrate than the smartphone. Some researchers have proposed client side algorithms that produce resource fairness [21, 30, 7] while other propose network assisted approaches that produce resource fairness [40, 5, 58]. Other researchers consider fairness issues between heterogeneous clients with different screen resolutions and attempt to provide clients with QoE fairness instead of resource fairness [16, 10].

The objective of this work is to propose an adaptation scheme that balances the need between

(1) QoE within a single streaming session and (2) QoE fairness across multiple streaming sessions considering the following: (1) buffer levels, (2) screen resolutions, (3) request quality, and (4) server bandwidth. To the best of our knowledge, there has not been work on both intra-session QoE and inter-session fairness that considers all these factors. For this reason, we conducted a systematic analysis of QoE and fairness in DASH and a comparison of adaptation algorithms using a rate-based algorithm, a buffer-based algorithm, a hybrid algorithm, a resource fairness algorithm (FESTIVE), and a QoE fairness algorithm (resolution-based). We compared these algorithms using static and dynamic bandwidth limitations with a single client and multiple clients competing for bandwidth. The key observations of this analysis and comparison are as follows:

1. The minimum bandwidth needed to stream at a quality level is on average 6.5% higher than the average bitrate listed for that quality level in the DASH MPD file. We refer to this as "the overhead of DASH".

2. The rate-based algorithm was able to achieve a low number of quality switches and high utilization at the cost of long and unstable initial phases due to the harmonic mean window not being fully populated during the initial phase.

3. The buffer-based algorithm was able to achieve high buffer levels due to its ability to adapt to small network changes however, this also produced a high number of quality switches.

4. The hybrid algorithm was able to achieve short and stable initial phases at the cost of low visual quality during the initial phase.

5. The resource fairness algorithm, FESTIVE, was able to achieve resource fairness and high buffer levels at the cost of low utilization under homogeneous client scenarios.

6. The QoE fairness algorithm, resolution-based, was able to achieve QoE fairness,

high utilization, high quality level, and a low number of quality switches under heterogeneous client scenarios.

7. The midway quality strategy balances the need for a short and stable initial phase that has a high visual quality by providing clients with a quality levels that is half of their fair share of the servers bandwidth.

Based on these observations we define the following design objectives of the new quality adaptation scheme: (1) provide QoE within individual streaming sessions using clients quality level, buffer level, frequency of quality switches, and magnitude of quality switches and, (2) to achieve QoE fairness across multiple streaming sessions using clients resolutions and SSIM values. We then propose FairQ, a client-server solution that controls the quality selection of a DASH session and coordinates fairness across streaming sessions while providing QoE enhancement for individual sessions.

The contributions of this thesis are as follows: (1) A survey of DASH adaptation algorithms and techniques. (2) As analysis of DASH techniques including the demonstration of the 6.5% average overhead of DASH. (3) A comparison of DASH algorithms and the impact of various dynamic network conditions on algorithm performance. (4) The proposal of FairQ, a client-server adaptation algorithm. (5) The evaluation of FairQ against a rate-based, buffer-based, hybrid, resource fairness, and QoE fairness algorithm under a variety of network and client scenarios.

The organization of this thesis is as follows. Chapter 2 presents a background of DASH and presents and overview of DASH adaptation schemes. Chapter 3 presents an analysis of QoE and fairness in DASH using a rate-based, buffer-based, hybrid, resource fairness (FESTIVE), and QoE fairness (resolution-based) algorithm under static and dynamic bandwidth limitations. Chapter 4 presents a comprehensive comparison of the algorithms used in Chapter 3 using a more network and client scenarios. Chapter 5 presents FairQ, our client-server adaptation scheme based on the key observations from Chapter 3 and 4. Chapter 6 evaluates FairQ against a rate-based algorithm, a buffer-based algorithm as presented in [23], a hybrid algorithm that is default algorithm in DASH

player GPAC [27], a fair algorithm as presented in [21], and a resolution aware algorithm under a range of network and client scenarios. Chapter 7 concludes the work with a summary and suggestions for future work.

Chapter 2

Background and Related Work

When setting up a DASH server, each video is transcoded into a predefined set of quality levels. Each quality level is quantified using its bitrate (bps) to indicate the amount of bandwidth required for transmission of that video. The video at each quality level is broken into segments representing a fixed playback duration (*e.g.*, 2 seconds) as shown in Fig. 2.1. Since all segments have the same playback duration, the boundaries of the same segment at different quality levels are all aligned. Thus, the streaming session can switch quality levels from one segment to the next. Information about quality levels and individual segments are summarized in the Media Presentation Description (MPD) file offered in XML format by the video server.

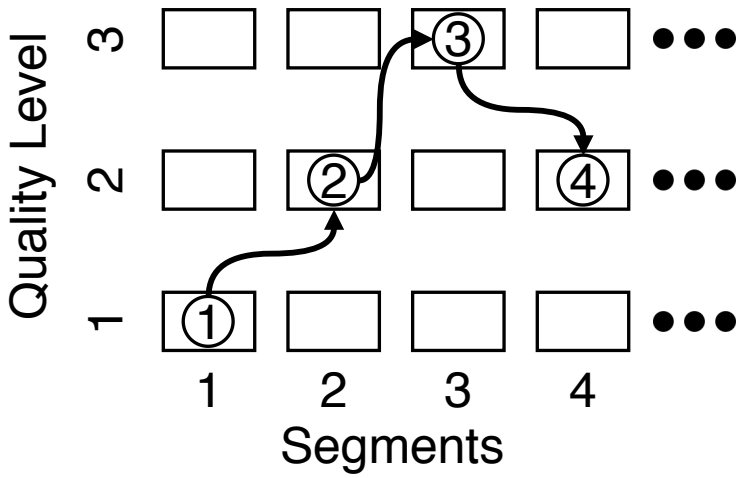

Figure 2.1: A DASH client dynamically adjusting the video bitrate during a streaming session.

To begin a DASH streaming session, the client first downloads the MPD file for the video of interest. It then requests segments in sequential order at a specific quality level according to various factors (*e.g.*, observed download rate of previous segments and the number of segment available in the playback buffer). This leads to a video stream with a dynamic quality as shown in Fig. 2.1.

The playback of the video is not started until there are sufficient segments in the playback buffer in order to ensure smooth playback regardless of the bandwidth and delay jitter. The time needed to fill the buffer with sufficient segments is referred to as the initial delay. The time between the first segment request and when video playback begins is referred to as the initial phase. Once the playback starts, the client enters the adaptation phase.

```
<Representation id="480x360 178.0kbps" mimeType="video/mp4"
                codecs="avc1.42c015" width="480" height="360"
                frameRate="24" sar="1:1" startWithSAP="1" bandwidth="178351" />

<Representation id="480x360 222.0kbps" mimeType="video/mp4"
                codecs="avc1.42c015" width="480" height="360"
                frameRate="24" sar="1:1" startWithSAP="1" bandwidth="221600" />

<Representation id="480x360 263.0kbps" mimeType="video/mp4"
                codecs="avc1.42c015" width="480" height="360"
                frameRate="24" sar="1:1" startWithSAP="1" bandwidth="262537" />

<Representation id="480x360 334.0kbps" mimeType="video/mp4"
                codecs="avc1.42c015" width="480" height="360"
                frameRate="24" sar="1:1" startWithSAP="1" bandwidth="334349" />

<Representation id="480x360 396.0kbps" mimeType="video/mp4"
                codecs="avc1.42c015" width="480" height="360"
                frameRate="24" sar="1:1" startWithSAP="1" bandwidth="396126" />
```

Figure 2.2: Subset of a MPD file for the video “Big Buck Bunny” from dataset in [28]

An example MPD file is shown in Fig. 2.2 that shows a video offered in 5 different quality levels, 178 Kbps, 222 Kbps, 263 Kbps, 334 Kbps, and 396 Kbps as indicated by the **id** field. The **bandwidth** field is used by clients to determine if their network can currently support that quality level and is simply the exact bitrate in bits per second instead if kilobits per second. To begin a streaming session a client would first download this MPD file to determine which quality levels are being offered. Typically, video players begin by choosing the lowest quality level available for the first segment because there is no useful buffer information or network information available at that time. The client can then calculate the download rate of the first segment, monitor the change in buffer level, or some other strategy to determine which segment to request next. This process repeats throughout the entire streaming session with the client taking periodical measurements and

requesting segments based on these measurements. What and how the client measures is an active area of research and the following section will give an overview of different strategies that have been developed.

2.1 Overview of DASH Adaptation Algorithms

Within a single streaming session, playback smoothness (the number of stalls during playback), visual stability (the number of quality-level switches) and the visual quality (the quality level of individual segments) are essential for good QoE [17, 47]. All three QoE metrics are highly impacted by the selection of the quality level of individual segments based on the estimated bandwidth, the buffer level, or some other metric. One of the simplest adaptation algorithms is to use the download rate of the most recently downloaded segment as a bandwidth estimation that is used to determine the quality of the next segment to be requests. This approach has a tendency to produce a large number of quality switches due to momentary changes in network conditions. This has the overall result of lowering the visual stability and the overall QoE of the video stream. Thus, the general approach to bandwidth estimation is to observe the throughput over multiple segments. By using multiple throughput observations we can create a more stable bandwidth estimation that ultimately leads to higher visual stability and higher overall QoE of the video stream. These algorithms are generally referred to as rate-based algorithms as they make quality decisions based on segment download rates (throughput observations). In this section, we will review different categories of DASH adaptation algorithms.

2.1.1 Rate-based Adaptation (client side)

Rate-based algorithms use throughput measurements of the previous n segments as bandwidth estimation. The value of n can be a fixed value [21, 22] or dynamically determined by the buffer level [13]. More specifically, in [13], $n = \frac{\text{Buffer Level}}{\text{Segment Duration}}$, thus, as the buffer level falls, n becomes smaller and makes the bandwidth estimation more sensitive to network changes. This reduces the

chances that the client is overestimating network conditions and allows the client to raise their buffer level back to a healthy range. Jiang *et al.* utilized the harmonic mean of the previous 20 throughput measurements as a bandwidth estimation [21]. The harmonic mean is less affected by outliers and is more stable than the arithmetic mean. Juluri *et al.* made the observation that segment sizes vary significantly under a given bitrate and propose a Segment Aware Rate Adaptation Algorithm (SARA) based on this observation [22]. The bandwidth estimation in SARA is a weighted harmonic mean of the download rate of the last n segments where the weight of each segment is proportional to its size. Lin *et al.* proposed a more accurate way for DASH clients to estimate their current bandwidth based on a metric called the Coefficient of Variance (CV) [31]. The CV of a segment is equal to the standard deviation divided by the average of the throughput of the last n segments. Larger CV values are associated with persistent bandwidth changes and smaller CV values are associated with momentary non-persistent bandwidth changes. Using this CV value allows clients to lower the number of quality switches and ultimately increase visual stability and overall QoE.

2.1.2 Buffer-based Adaptation (client or server side)

Another general approach is to select segment qualities with the goal of maintaining a steady buffer level throughout the streaming session. Huang *et al.* proposed an adaptation algorithm that only uses throughput values in the initial phase and relies solely on buffer level for segment selection in the adaptation phase [18]. Le *et al.* utilized the current buffer level and previous segment quality to estimate future buffer levels allowing the client to balance the need between high quality video segments and buffer stability [26]. This involves predicting the segment quality and buffer level for the next N segments where the value of N depends on the buffer level. This allows the client to react to momentary network changes when the buffer level is low. Sani *et al.* proposed QoE-aware model of the relationship between video quality and buffer state changes [49]. This algorithm does not rely on a heuristic to model the relationship between video quality and buffer occupancy making it more stable than traditional buffer based approaches. Spiteri *et al.* formulated bitrate

adaptation as a utility maximization problem and devise an online buffer based control algorithm called BOLA [52]. Like the algorithm presented in [49], BOLA is not based on heuristics and avoids the overhead of complex bandwidth estimation techniques.

Liu *et al*. made the observation that in order to avoid video freezing during playback, a clients buffer must not deplete at a higher rate than it is filled [32]. To ensure that does not happen, Liu *et al*. proposed that the segment fetch time (SFT), which is defined as the time between the GET request for that segment and when the last bit of the segment was received, must be at least slightly smaller than the media segment duration (MSD). Further, they define two values, a switch up and a switch down value, such that if the ratio $\mu = \frac{MSD}{SFT}$ falls below the switch down value or above the switch up value the client will request segments of lower and higher qualities respectively. Kim *et al*. took this approach a step further by proposing a server-side adaptation scheme that estimates client buffer levels based on the length of time between subsequent segment requests [23]. This approach is very similar to to approach taken in [32] however, all the adaptation logic is placed on the server side. The algorithm presented in [23] is shown in Fig. 2.3 as it will be used as a comparison algorithm during evaluation. Where λ is the low buffer threshold in seconds, G_{UP}, G_{DOWN}, and β are scaling constants. The values for these are set to the default values listed in [23].

2.1.3 Hybrid of Rate-based and Buffer-based Adaptation (client or server side)

A third general approach, referred to as the hybrid approach, is to utilize both buffer information and throughput measurements to make quality decisions. Menkovski *et al*. proposed an intelligent streaming client capable of learning an optimal control strategy based on a subjective QoE measure [34]. Their QoE measure is the weighted average of three variables, s, f, and c, where s is the degradation of video quality due to bitrate restriction, f is the degradation of video quality due to stalls in video playback, and c is the degradation of video quality due to quality switches. Claeys *et al*. took a similar approach to [34] and propose a reinforcement learning based DASH client

Algorithm 1: Buffer-based Algorithm

$G_{Up} = G_{Down} = 150000$;
$\lambda = 10$ $\beta = 0.25$;
B_{k-2} = buffer level at k-2;
B_{k-1} = buffer level at k-1;
L_{k-1} = last segment length in seconds;
T_{k-2} = Last segment upload start time;
bitrate;
Function `on_segment_upload(`*segment*`)`**:**
    Δ_{k-1} = now() - T_{k-2};
    B_k = max(0, B_{k-2} + (L_{k-1}- Δ_{k-1}));
    $B_{k-2} = B_{k-1}$;
    $B_{k-1} = B_k$;
    $E_{k-1} = B_{k-1}$- B_{k-2};
    if $B_k > \lambda$ **then**
        $E_{k-1} = E_{k-1} + \beta \times L_{k-1}$;
    if $E_{k-1} > 0$ **then**
        bitrate = bitrate + $G_{Up} \times E_{k-1}$;
    else if $E_{k-1} < 0$ **then**
        bitrate = bitrate + $G_{Down} \times E_{k-1}$;
    T_{k-2} = now();
    L_{k-1} = segment.length;

Figure 2.3: Pseudo code for buffer based algorithm presented in [23]

based on a subjective QoE measure [9]. The three metrics that make up the subjective QoE metric are the segment quality, number of quality switches, and the number of and length of video stalls due to an empty buffer.

Chiariotti *et al.* took a similair approach to [9] with a Reinforcement Learning, online DASH controller based on video quality, quality switches, and stalls in video playback [6]. Their DASH controller out performs the algorithm presented in [9]. Zhou *et al.* proposed a Markov-decision-based rate adaption approach for DASH called mDASH that utilizes video quality, frequency and size of quality switches, stalls in video playback, and buffer levels [60]. Based on their analysis and optimization problem formulation, they also proposed a sub-optimal greedy algorithm to reduce complexity and support real time streaming. Li *et al.* propose an adaptation algorithm for DASH built off of mDASH that improves the start up time and the video rate switching performance of mDASH[29]. This was done by utilizing a different start up algorithm that begins on a low bitrate and gradually ramps up instead of making large quality switches.

An example of a hybrid algorithm in practice is the default algorithm in the GPAC DASH player [27]. GPAC is an open source multimedia framework that is used for research, academic purposes, and even by industry. The default algorithm uses the download rate of the last segment, the current buffer level, and the previous buffer level to choose segment qualities. It is somewhat unique in that it only uses the download rate of the last segment when switching to a lower quality level. When switching up, the decision is based on buffer level and only switches up one quality level. The algorithm is shown Fig. 2.4 as it will be used as a comparison algorithm during evaluation. For this algorithm, the high buffer threshold (buf_high) is set to $m - l$ and the low buffer threshold (buf_low) is set to l where m is the maximum buffer length and l is the segment length in seconds.

In summary, towards improving QoE, researches have proposed rate-based algorithms, buffer-based algorithms, and hybrid algorithms. Rate-based algorithms use throughput measurements to determine what quality of segments should be selected by clients. This can be done using the harmonic mean of throughput measurements [7], a weighted average based on segments sizes in

Algorithm 2: Hybrid Algorithm

```
1  rate = download rate of last segment;
2  switch = TRUE;
3  control = current buffer level - previous buffer level;
4  go_up = FALSE;
5  target_quality = current quality;
6  if current buffer level > maximum buffer level then
7      control = 1;
8  if current buffer level < buf_low then
9      go_up = FALSE;
10     if current buffer level = 0 then
11         rate = lowest quality level;
12     else
13         rate = current quality − 10;
14 else if control > 0 and current buffer level > buf_high then
15     go_up = TRUE;
16 else
17     switch = FALSE;
18 if switch == TRUE then
19     if go_up == TRUE then
20         target_quality = lowest quality above current quality;
21     else
22         target_quality = highest quality below rate;
23 segs_since_switch = segs_since_switch + 1;
24 if segs_since_switch > 1 then
25     switch to target_quality;
26     segs_since_switch = 0;
```

Figure 2.4: Pseudo code default GPAC algorithm

bytes [22], or the standard deviation of throughput measurements [31]. Buffer-based algorithms use the buffer level and the change in buffer level to determine what quality of segments should be selected by clients. This can be done by comparing the segment download time to the length of the segment in seconds on the client [32] and the server [23], the current buffer level and the quality level of the previous segment [26], or as a maximization problem based on buffer levels [52]. Hybrid algorithms utilize some combination of throughput measurements and buffer information to determine what quality of segments should be selected by clients. This is generally done by creating a QoE measure based on buffer level, segment quality, and quality switches and attempting to maximize it [34, 9, 60].

2.2 Fairness Issues across Multiple DASH Sessions

The algorithms presented thus far are susceptible to fairness issues that can arise when multiple clients are streaming simultaneously and competing for bandwidth as shown in [2]. These issues arise due to the ON-OFF nature of segment downloads where clients switch between downloading segments (ON period) and inactivity (OFF periods). The length of the ON-OFF cycle varies depending on the frequency of segment requests. If all clients are downloading segments asynchronously, as shown in Fig. 2.5, this can lead to clients overestimating available bandwidth. If all clients are downloading segments synchronously, as shown in Fig. 2.6, this can lead to clients underestimating available bandwidth. In both cases, there is potential for unfair division of network resources, leading to stalls, lower visual stability, and degraded visual quality. Here, we review the client-side and network solutions to fairness issues, as well as QoE fairness algorithms.

2.2.1 Client-side solutions

To address these fairness issues, there are client-side solutions and network solutions. On the client side, Zahran *et al.* solved this by scaling bandwidth estimations based on buffer levels with high buffer levels slightly increasing the bandwidth estimation and low buffer levels slightly decreasing

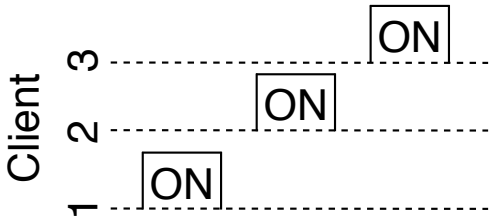

Figure 2.5: Unsynchronized segment downloads.

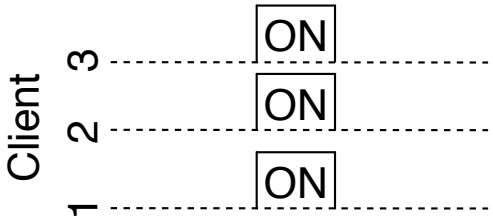

Figure 2.6: Synchronized segment downloads.

the throughput estimation [59]. Scaling the bandwidth estimation of low buffer clients prevents those clients from requesting high quality segments that could lower their buffer level even further. Scaling the bandwidth estimation of high buffer clients prevents them from filling their buffer levels and entering periods of activity which helps eliminate the issues described in [2]. These solutions all decrease the chances that clients will get stuck in a cycle of either downloading segments at the same time or downloading segments alone.

Li *et al.* proposed a control algorithm that allows clients to request segments at an increasing rate until the observed throughput drops, similar to TCP Slow Start [30]. De Cicco *et al.* solved this by allowing clients to continue requesting segments if their buffer levels are higher than the maximum buffer level unless they are at the highest quality level [7]. Jiang *et al.* proposed a randomized requesting scheme that has clients calculate a new target buffer level on each segment request in the range $(m-d, m+d]$ where d is the segment playback time in seconds and m is the maximum buffer size in seconds [21]. Clients are then configured to request segments if the current buffer level is below the newly calculated target buffer level. The algorithm in [21], called

FESTIVE, is shown in Fig. 2.7 as it will be used as a comparison algorithm during evaluation.

Algorithm 3: FESTIVE Algorithm

```
p_val = 0.85 * harmonic mean value;
cur_score = 0;
ref_score = 0;
ref_rate = cur_rate;
if cur_rate > p_val then
    ref_rate = one quality level lower than current;
else if harmonic mean value > one quality level higher than current then
    if segs_since_switch > current quality level then
        ref_rate = one quality level higher than current ;
calculate score for current quality;
calculate score for reference quality;
if cur_score > ref_score then
    if not in initial phase then
        switch to ref_rate;
        segs_since_switch = 0;
```

Figure 2.7: Pseudo code for FESTIVE algorithm presented in [21]

FESTIVE utilizes the harmonic mean of the last 20 segment download rates to obtain the harmonic mean value in the above algorithm. To calculate the score for the current and reference quality respectively the following formulas are used (n is the number of quality switches in the last 20 segments, and *hmean* is the harmonic mean value):

$$\text{cur\_score} = 2^n + 12 \times \left| \frac{\text{cur\_rate}}{min(hmean, ref\_rate)} - 1 \right|$$

$$\text{ref\_score} = (2^n + 1) + 12 \times \left| \frac{\text{ref\_rate}}{min(hmean, ref\_rate)} - 1 \right|$$

Notice that FESTIVE takes a very conservative approach to increasing quality levels by prohibiting quality jumps larger than one level and only switching up after receiving as many segments as the current quality level. For example to switch from quality level n to $n+1$ a client would need

to have received n continuous segments at a quality level of n. This makes it more difficult for higher quality clients to increase their quality level and prevents them from using more than their fair share of available bandwidth. This coupled with the randomized requesting scheme has the overall affect of increasing fairness between clients.

2.2.2 Network Solutions

One network approach is to have some entity other than the client estimate network conditions. This estimation can then be used by all clients in the network resulting in fairness due to clients all using the same bandwidth estimation. This is the approach taken by Mok *et al.* when designing QDASH [36]. QDASH is composed of two separate components, QDASH-abw, and QDASH-qoe. QDASH-abw is a measurement proxy placed between the client and the server that probes the network to determine the current network capacity. QDASH-qoe is the adaptation algorithm present on the client side that determines segment quality levels based on information gathered by QDASH-abw. Another network approach is to have some entity other than the client not only estimate network conditions but also determine client quality levels.

El Essaili *et al.* took this approach via a QoE optimizer for wireless networks that determines the optimal rate for streaming clients based on network conditions [14]. The optimal rate determined by the optimizer is then used by a QoE proxy that rewrites clients HTTP requests unbeknownst to the client and the server. Chen *et al.* proposed a scheduling framework for adaptive video delivery of cellular networks called AVIS [5]. AVIS attempts to provide clients with (1) a fair allocation of network resources, (2) a stable bitrate for each user, and (3) efficient use of network resources. Petrangeli *et al.* proposed a reinforcement learning coordination proxy based algorithm that achieved a 60% increase in fairness over Microsofts Smoothing Streaming algorithm [40]. These fairness solutions all address fairness issues that arise when multiple clients share a bottleneck within the network by providing these clients an equal share of the bottleneck. We think this notion of fairness relies on an overly simplified homogeneous network scenario where each client has identical bandwidth requirements.

2.2.3 QoE Fairness Algorithms

A more realistic network scenario would involve an array of heterogeneous clients with varying bandwidth requirements. Under this more realistic network scenario, giving each client an equal share of network resources would lead to under serving high bandwidth clients and over serving low bandwidth clients ultimately leading to a low QoE for both. To this end, Petrangeli *et al.* proposed FINEAS, an in network heuristic algorithm to achieve QoE across a group of heterogeneous clients [41]. This algorithm consists of a client side algorithm based on QoE measures (video quality, video stability, etc) and a fairness signal that is computed by system of coordination proxies. The client uses the fairness signal as a basis for its quality selection however, it also considers its own QoE. This results in a video quality that may achieve QoE fairness without each client streaming at the same quality level. Yan *et al.* presented a hybrid edge cloud and client adaptation algorithm for HTTP adaptive streaming in cellular networks called Prius [58]. Prius attempts to maximize fairness across multiple streaming sessions by reducing the variation in QoE across client and not resources.

Both Georgopoulos *et al.* and Cofano *et al.* utilized the Structural Similarity Index (SSIM) in order to achieve QoE fairness in contrast to resource fairness [16, 10]. This is achieved by solving an optimization problem that maximizes the minimum SSIM over all clients which may result in a solution in which QoE is fair across clients but resources are not. Przylucki *et al.* took a similiar approach to [16] and [10] by maximizing the minimum QoE measure over all clients however, their QoE measure is not based on SSIM [42]. The QoE they use is based on SVC videos and the ratio between the enhancement layers currently being used and all possible enhancement layers at a given resolution. Ramakrishnan *et al.* formulated a multi-client bandwidth allocation problem within the convex optimization framework that attempts to achieve QoE fairness instead of just resource fairness [46]. The convex optimization framework allows the use of several different QoE measurements, such as PSRN or SSIM, and also considers client buffer levels in order to achieve QoE fairness.

The algorithm presented in [10] attempts to maximize $[min_{l_n \in L_n} U_n(l_n)]$ on the server side side subject to $\sum_{n=1}^{N} l_n < C$. N is the number of clients, C is the servers bandwidth, l_n is the bitrate for client n, L_n is a set of all possible bitrates, and U_n is a utility function that associates each bitrate in L_n to its perceived video quality. U_n is simply the SSIM value for a video at a given bitrate and a given resolution. They provide a simple heuristic algorithm where each client is initially set to the lowest possible bitrate. At each round, client n is selected with the lowest value of $U_n(l_n)$ and has its quality level raised by one. This is repeated until there is not enough bandwidth to raise the quality level of client n. A variation of the algorithm presented in [10] will be used in evaluation.

Algorithm 4: Resolution-Based Algorithm

```
1 foreach client i do
2 |   q_i = lowest quality level ;
3 end
4 while ∑_i q_i < C do
5 |   select client k with the lowest SSIM value ;
6 |   raise q_k by one level ;
7 end
```

Figure 2.8: Pseudo code for variation of algorithm presented in [10]

The variation of the algorithm presented in [10] is shown in Fig. 2.8 where C is the servers bandwidth, q_i is the final quality of client i, and r_i is the request quality of client i. The clients request quality, r_i, is determined by using the harmonic mean of the previous 20 throughput values as a bandwidth estimation. The algorithm is almost identical to [10] in that each client is initially set to the lowest quality level. At each round some client i is selected that has the lowest SSIM value and its quality level is raised by one. The variation is that the Algorithm shown in 2.8 returns to the client $min(r_i, q_i)$. This allows the algorithm to adapt to changes in network conditions experienced by the client. The solution initially calculated by the algorithm need only be calculated whenever a new client joins or leaves and does not need to be calculated upon every segment request as it is based on the constant resolution of each client.

Some researches attempt to solve more fundamental issue with DASH, such as the limitation

of discrete quality levels, the overhead of transcoding a large number of quality levels, or transcoding multiple quality levels of live streams in time. To address the limitation of discrete quality levels, Huang *et al*. proposed a cloud based SVC proxy capable of transcoding videos to better meet network conditions. To achieve this in real-time, the proxy must first decode the video to be streamed into some intermediate quality and then encode the video to SVC utilizing a cluster of cloud computers [19]. Pu *et al*. proposed WiDASH, an adaptation scheme designed to optimize quality levels across multiple wireless clients [43]. This is accomplished via a proxy placed between the internet and the core wireless network that is capable of transcoding quality levels that more accurately match clients available bandwidth.

To lower the overhead of DASH transcoding, Krishnappa *et al*. proposed a hybrid transcoding approach where a DASH server only stores the first segment of each quality level being offered with the other segments being transcoded in real time if requested [24]. To help transcode multiple quality levels of live content, Aparicio-Pardo *et al*. proposed a heuristic algorithm that utilizes the cloud [4]. The algorithm iterates through channels in decreasing order of popularity and for each channel, a CPU budget is decided and finally a set of representations is decided based on the assigned CPU budget. Ma *et al*. proposed a dynamic scheduling algorithm for video transcoding in the cloud with a focus on preparing several quality levels for user generated content such as a Youtube video [33]. This algorithm utilizes two video segment queues, a higher priority queue and a low priority queue. Initially all segments to be prepared are placed in the low priority queue and a segment is moved to the high priority queue when it has been requested by a client along with all subsequent segments at that quality level.

In summary, Akhshabi *et al*. reveal fairness issues that can arise when multiple clients are streaming simultaneously and competition for bandwidth [2]. Researchers proposed client side solutions to this problem [21, 7, 30, 59] and network solutions [36, 14, 5, 40]. These solutions fix the issues presented in [2] by attempting to allocate network resources equally among clients. Some researchers argue that this is an overly simple notion of fairness as different clients may

have different bandwidth needs and create algorithms that attempt to achieve QoE fairness instead of just resource fairness [16, 10, 41, 58]. Other researchers focus on more fundamental issues with DASH, such as the limitation of discrete quality levels [19, 43], the overhead of transcoding a large number of quality levels [24], or transcoding multiple quality levels of live streams in time [4].

Overall, the adaptation algorithms are concerned with two primary design objectives: (1) QoE within a single streaming session (such as buffer level, segment quality, visual stability, and bandwidth utilization), and (2) fairness across multiple streaming sessions. To the best of our knowledge, there has not been an adaptation scheme that considers all the factors within a streaming session that affect QoE as well as utilizing server side intervention to enforce QoE fairness. In this paper, we propose FairQ, a client-server adaptation scheme that considers both QoE within a single streaming session and fairness across multiple streaming sessions. The following section will be a systematic analysis of QoE and fairness of DASH.

Chapter 3

Analysis of QoE and Fairness in DASH

Towards improving QoE in DASH, researchers have utilized buffer levels [18, 23], segment sizes in bytes [18, 22], the harmonic mean of throughput values [7, 30], and the ratio between segment length and download time [56, 32] to determine which segment quality a client should choose. Researchers have also designed adaptation algorithms that consider fairness between simultaneously streaming clients [21, 30, 7]. In summary, these streaming systems are concerned with two primary design objectives: (1) fairness across multiple streaming sessions and (2) factors within a single streaming session (such as buffer level, segment quality, visual stability, and bandwidth utilization) that ultimately impact the users QoE.

Akhshabi *et al.* reveal fairness issues that arise when two or more adaptive media players stream content while sharing the same bottleneck [2]. The two extreme cases are that (1) a group of two or more clients often download segments at the same time leading them to under estimate network conditions or (2) clients often download segments alone, leading them to over estimate network conditions. Jiang *et al.* solve this by using a randomized approach to requesting segments that decreases the chances of clients always requesting segments at the same time [21]. Li *et al.* solve this by ramping up the rate of segment requests for each client until the observed throughput decreases [30]. De Cicco *et al.* solve this by allowing clients to continuing requesting segments if their buffer levels are higher than the maximum buffer level unless they are at the highest quality level [7].

Factors within a single streaming session that ultimately impact the users QoE include buffer level, segment quality, visual stability, and bandwidth utilization. Jiang *et al.* and Juluri *et al.* both utilize a harmonic mean of previous throughput values to increase visual stability and bandwidth utilization [21, 22]. Huang *et al.* and Kim *et al.* both utilize buffer-based algorithms that pick

segment qualities based on the current buffer level and how fast it is changing in an attempt to increase buffer levels [18, 23]. Dubin *et al.* use the median of the last n throughput values where $n = \frac{BufferCapacity}{SegmentDuration}$ to increase visual stability, bandwidth utilization, and buffer levels [13].

In this thesis, we are interested in balancing the quality across multiple sessions while enhancing QoE within individual sessions. Towards this objective, Jiang *et al.* utilize a randomized scheduler, a tunable, stateful bit rate selection technique that allows for a trade off between visual quality and visual stability, and a harmonic mean of the last 20 throughput values [21]. Li *et al.* utilize a probe and adapt method, similar to TCPs congestion control, that increases the rate of segment requests until the client experiences congestion, and an exponentially weighted moving average of throughput values [30]. De Cicco *et al.* utilize a harmonic mean of the last 5 throughput values, and allow clients to continuing requesting segments if their buffer levels are higher than the maximum buffer level unless they are at the highest quality level [7].

In contrast to these works, we propose FairQ, a new quality adaptation scheme that utilizes server side info (such as currently available bandwidth and number of connected clients), client side info (such as buffer level, segment quality, and download rate), and network conditions. The design of FairQ is inspired by a systematic analysis on the impact of fairness, bandwidth utilization, buffer level, visual stability, and visual quality on QoE under various controlled emulated network conditions. By considering how these factors interact with each other and quantify their impacts on the QoE, FairQ balances the quality across streaming sessions while enhancing quality within individual sessions. The performance metrics used during this analysis are:

- **Buffer Level**: The buffer level in seconds over time for a single client. We can also calculate an average buffer level for a client over a streaming session. Buffer levels, or average buffer levels, can help determine how susceptible a client is to stalls in video playback. Rodrguez *et al.* suggest that stalls in video playback, or rebuffering events, are shown to negatively affect a users QoE by [47].

- **Segment Quality**: The quality level (in Kbps or Mbps) of segments that a client

requests over time. We can also calculate an average segment quality for a client over a streaming session. Segment quality, or average segment quality, can help determine the visual quality of a video stream.

- **Download Rate**: The download rate (in Kbps or Mbps) of video segments that a client requests over time. We can also calculate an average download rate for a client over a streaming session. Download rate, or average download rate, can help determine the network conditions for a client during a streaming session.

- **Number of Quality Switches**: The number of times that segment quality changes for a single client over time. We can also calculate an average number of quality switches over multiple runs of the same test. The number of quality switches, or average number of quality switches, can help determine the visual stability of a video stream. Mok *et al*. and Rodrguez *et al*. both suggest that low visual stability negatively affects a users QoE [47] and [37].

- **Standard Deviation of Quality Levels Between Clients**: The standard deviation of segment quality levels between clients over time. This is calculated on a per second basis. We can also calculate an average standard deviation of quality levels between clients over a single streaming session. In this chapter, we use homogeneous client scenarios (clients have the same screen resolutions), thus, to achieve fairness each client should stream at the same quality level. For this reason, we consider the standard deviation of quality levels between clients to be an appropriate measure of fairness. This will be referred to as "Fairness" in tables throughout this chapter.

- **Initial Phase Length**: The length of time, in seconds, between a clients first segment request and when video playback starts. We can also calculate average initial phase length over multiple runs of the same test. Initial phase length, or average

initial phase length, can be used to determine the clients QoE as Mok *et al*. suggest that long initial phase lengths contribute negatively to a users QoE [37]. This will be referred to as "I.P. Length" in tables throughout this chapter.

- **Bandwidth Utilization**: The ratio between total segment quality and available bandwidth over time. This is calculated using average segment quality and average available bandwidth over a streaming session. We can also calculate average bandwidth utilization over several streaming sessions. Bandwidth utilization, or average bandwidth utilization, can help determine how accurately a streaming system is able to estimate available bandwidth. This ultimately determines segment quality.

The organization of this chapter will be as follows: Section 3.1 will describe our experimental set up and perform simple experiments to validate it. Section 3.2 will examine the overhead of DASH. Section 3.3 will perform a comparison of DASH algorithms under simple network scenarios. Section 3.4 will examine the pros and cons of different harmonic mean window sizes. Section 3.5 will compare different approaches during the initial phase. Section 3.6 will show a server and client side solution to fairness issues in DASH and Section 3.7 will be a summary.

3.1 Experimental Setup

We setup our testbed with three goals: First, we want to create a controllable environment that will allow us to conduct measurements in various network conditions while utilizing different adaptation schemes and number of clients. This goal requires us to have direct control of network settings, such as the bandwidth limit, and the latency of the network. This goal also requires us to have access to all necessary parameters of a DASH system, such as the adaptation algorithm being used, buffer size, statistics being logged, and whether clients should connect to a proxy or not. In order to have access to all the necessary DASH parameters an open source DASH client must be used.

Second, we need to create a wide range of testing scenarios to determine the fairness and QoE issues of different adaptation schemes. Similarly to the first goal, this goal also requires us to have direct control over network settings. This goal also requires us to use synthetic network scenarios that are designed to isolate specific variables we want to test. Finally, this goal requires us to utilize real-trace driven network conditions and a real DASH dataset in order to evaluate various streaming systems under realistic conditions.

Finally, the testbed must support repeatable experiments, which will enable us to conduct fair comparisons between various streaming systems under different settings. This goal requires us to have direct control over network settings so that we can reproduce the exact same network conditions between experiments. Providing these goals for our testbed, we chose to develop our own DASH emulation system. This includes a real host (server + client), real implementation of DASH clients, real traffic, and bandwidth emulation.

3.1.1 Setup

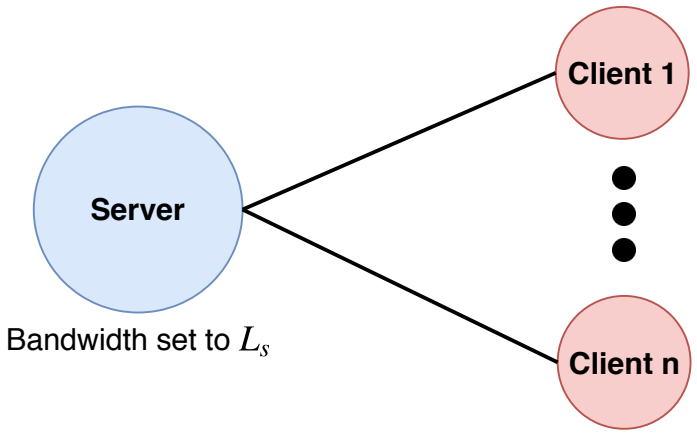

Figure 3.1: The default network setup where clients connect directly to the video server.

As shown in Figures 3.1 and 3.2, there are two experimental setups used referred to as default setting and proxy setting. The default setting represents a typical DASH setting where client requests are directly sent to the server and segments are streamed directly to clients. In the proxy

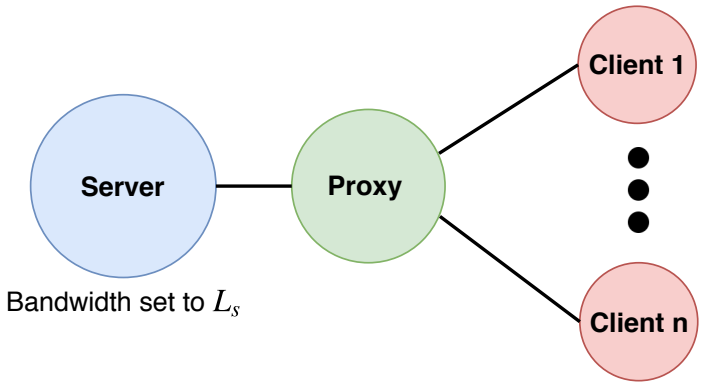

Figure 3.2: The proxy network setup where clients connect to a proxy placed between them and the video server.

setting, the communication between clients and server is tunneled through a proxy. The proxy setting allows the analysis of scenarios in which an entity other than the client is making quality decisions which increases the potential test cases. The video server and the proxy are each running on their own virtual machine which is hosted by Cybera at the University of Calgary. Each machine has 8GB of ram, 4 virtual CPUs, 40GB of storage with Ubuntu 14.04 LTS. Each client is implemented via a modified version of GPAC 0.6.1 that is using the harmonic mean of the last n (determined by the test being run) throughput values as the adaptation algorithm. GPAC is an open source multimedia framework that is used for research, academic purposes, and even by industry [27]. GPAC was chosen because it is open source which allows for greater control, it is widely used, and it provides DASH support. A buffer size of 30 seconds, as used by Jiang *et al.* and Li *et al.* [21] [30], was chosen and video playback does not start until the buffer level has reached 30 seconds. All clients are running on the same machine that has 8GB of ram, 2 CPUs, 250GB of storage with Ubuntu 14.04 LTS.

The dataset used was provided by Lederer *et al.* because it is a commonly used dataset for video streaming [28, 7, 22, 13, 61, 54]. This datatset offers 6 different videos with segment sizes ranging from 1 to 15 seconds The default segment size is set to 2 seconds because it is a common setting for video streaming [21, 7, 30]. We noticed that the actual number of bytes in a segment

varies from one segment to the next, which directly impacts the delivery time of each segment. Since DASH considers only the bandwidth field in the MPD file of to-be-streamed videos but not the content, we only need to have one video from the dataset that best represents the range of segment size in bytes. To do so, we analyzed the standard deviation of segment sizes for each video. This was done by calculating the standard deviation of segment sizes for each quality level and each video. These standard deviations were then averaged for each video and are shown below.

	Bunny	**Elephants**	**Forest**	**Red Bull**	**Swiss**	**Tears**	**Valkaama**	**Avg.**
Std. Dev.	93035	119564	80468	43219	58588	228554	60996	97775

Table 3.1: Average standard deviation of segment sizes for each video at a segment size of 2 seconds.

As shown in Table 3.1 the average standard deviation values range from 43219 to 228554 with an average of 97775. The ”Big Buck Bunny” video (”Bunny” for short) is the video with the standard deviation value closest to the overall standard deviation value which is why it was chosen for all DASH tests in this thesis. The quality levels offered at a segment size of 2 seconds for the video Bunny are shown in the table below.

Quality Level	**Resolution**	**Quality Level**	**Resolution**
4.2 Mbps	1920x1080	595 Kbps	854x480
3.8 Mbps	1920x1080	522 Kbps	854x480
3.5 Mbps	1920x1080	396 Kbps	480x360
3.1 Mbps	1920x1080	334 Kbps	480x360
2.5 Mbps	1920x1080	263 Kbps	480x360
2.1 Mbps	1920x1080	222 Kbps	480x360
1.5 Mbps	1280x720	178 Kbps	480x360
1.2 Mbps	1280x720	131 Kbps	320x240
1 Mbps	1280x720	89 Kbps	320x240
791 Kbps	1280x720	46 Kbps	320x240

Table 3.2: Quality Level Information for the “Big Buck Bunny” video at a segment length of 2 seconds from the dataset in [28]

For bandwidth emulation, we use `tc`, a Linux program for configuring packet scheduling. The choice to use `tc` was made due to the fact that `tc` is open source, widely used and tested, and has extensive documentation available. As shown in Fig. 3.3, `tc` controls traffic by using a

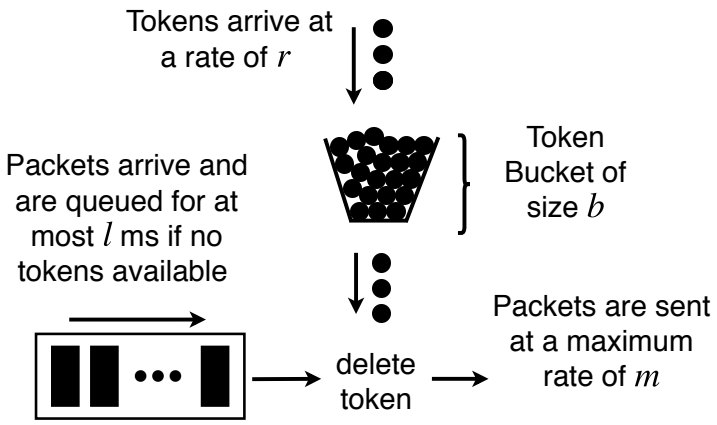

Figure 3.3: How the Linux tool `tc` controls bandwidth by associating each incoming packet with a token from the token bucket.

buffer, called a bucket, of size b. This bucket is constantly filled with tokens at a rate of r. Each token is associated with an arriving data packet from the data queue and is then deleted from the bucket when that packet is sent. The rate that packets arrive at is determined by the DASH clients requesting segments. If packets arrive and no tokens are available then these packets are queued for at most l ms before being dropped. If packets arrive and the token bucket is full then these packets are sent out at a rate of m which is the maximum rate that the bucket can be emptied at.

If b is sufficiently large and m is not set, then it is possible for bursts of packets to be sent at rates that exceed r. To prevent this, m can be set to the same value of r which is the default setting for this thesis. r will be set to the bandwidth limit required by the current test. b and l are the parameters that have the greatest effect on the behaviour of `tc`. If b is set too low then the rate at which packets are sent out may be much smaller than r. However, increasing b past this low threshold does not affect the results due to m being set to the same value as r. If l is set too low, then `tc` will drop a high number of packets resulting in these packets being resent by TCP. If this occurs frequently it will ultimately result in a low QoE for the user due to low bandwidth utilization. Thus it is imperative that b and especially l are set properly. The following section

contains analysis of `tc` that guide the selection of proper values for b and l.

3.1.2 Validating our Experimental Setup

As stated in the previous section, b and l have the greatest effect on the behaviour of `tc`. To determine appropriate values for b and l, three values for b were used, 12Kb, 120Kb, and 1.2Mb, and three values for l were used, 2ms, 20ms, and 200ms. The values for b start at the experimentally determined minimum (any size below 12Kb results in DASH clients being unable to stream due to timeout errors) and increase by a factor of 10. The values for l start at a value that is likely to result in many drops and increase by a factor of 10.

The combination of values for b and l resulted in a total of nine different settings, each of which were tested 10 times by streaming a DASH video for 6 minutes while limiting the bandwidth to 1.5 Mbps. The download rate of each segment of the video was measured by the client for each test and then averaged over 10 tests. It is likely that a video segment consists of several TCP packets however, the download rate is calculated on the application layer over the entire video segment and not for each individual packet. As a result, the download rate will be lower if `tc` drops any packets due to a low l value. The setting that produced the highest average download rate without going over the 1.5 Mbps limit was determined to be the optimum setting.

All the b values tested performed almost identically with l having the greatest affect on the behaviour of `tc`. A b value of 12 Kbits was chosen because it resulted in the highest average download rate of all the values tested. As shown in Fig. 3.4, the download rates of the segments are much lower than the bandwidth limit of 1.5 Mbps. This is due to the very small l of 2ms that causes `tc` to frequently drop packets. Increasing l by a factor of 10, to 20 ms, dramatically increases the observed download rate as shown in Fig. 3.5. However, there is still a drop in the download rate just before the 40th segment which is likely caused by `tc` dropping packets. Further increasing l by a factor of 10, to 200 ms, eliminated the drop in download rate as shown in Fig. 3.6. This setting ($l = 200$ ms and $b = 12$ Kbit) resulted in an average download rate of 1.44 Mbps and was determined to be the optimum setting and thus the default `tc` setting in this thesis.

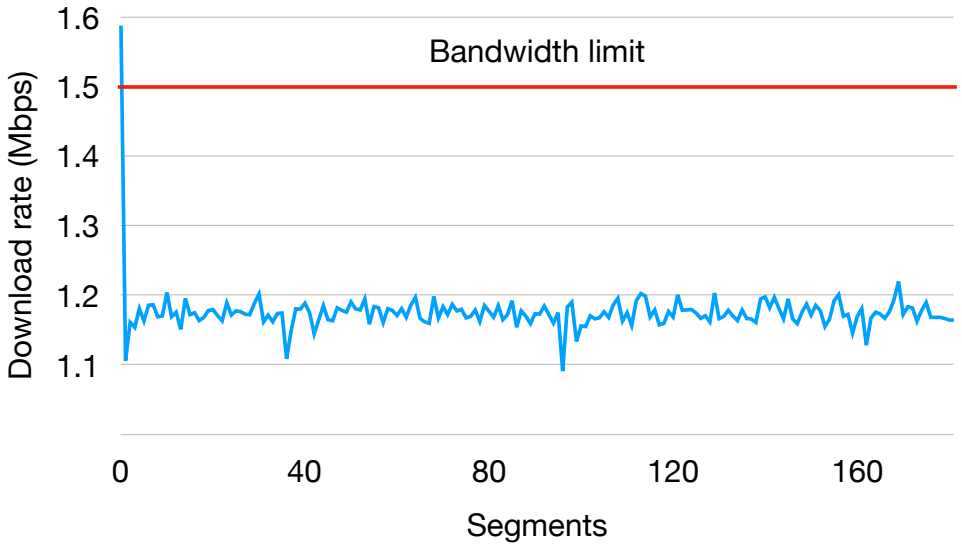

Figure 3.4: Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 2 ms respectively.

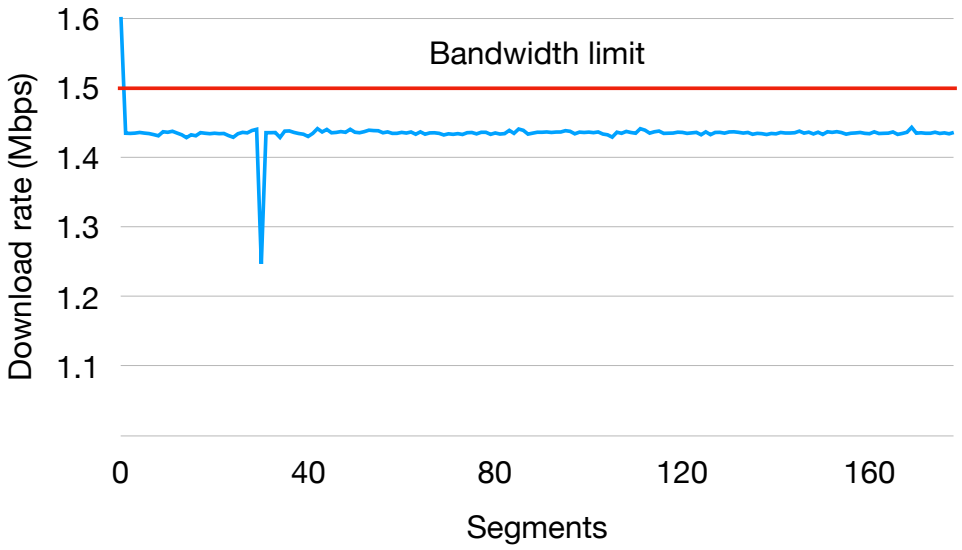

Figure 3.5: Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 20 ms respectively.

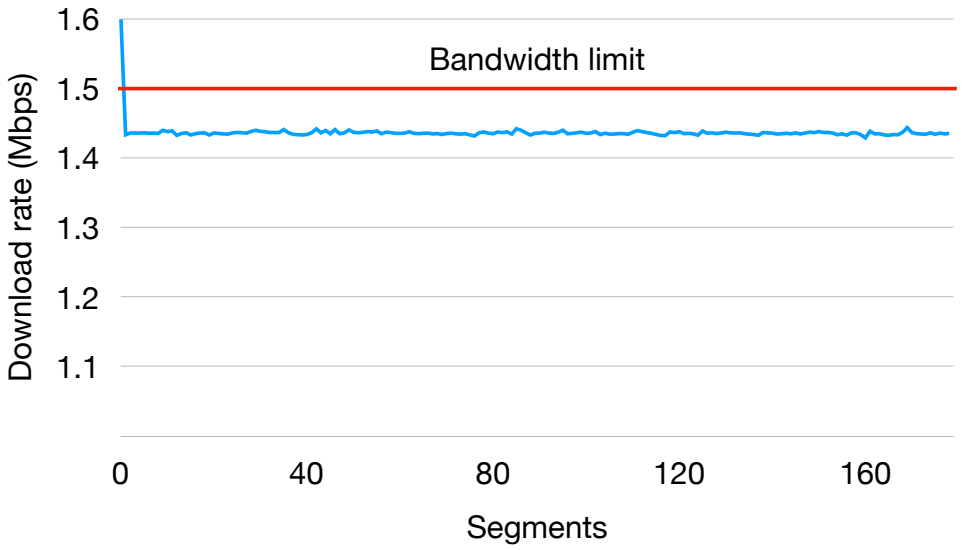

Figure 3.6: Download rate of segments compared to the bandwidth limit of 1.5 Mbps with `tc` parameters of b and l set to 12 Kbit and 200 ms respectively.

It is important to have a testbed that allows for a great degree of control, can produce a wide range of testing scenarios, and allows tests to be easily repeated. Towards these goals, we have presented our DASH emulation system that includes a real host, real implementation of DASH clients, real traffic, and bandwidth emulation. This system includes two test settings, the default setting where clients connect directly to the server, and the proxy setting where client requests are first tunnelled through a proxy before being sent to the server. We chose a video that best approximates the variation in segment size across all videos in the dataset as our default video. The bandwidth emulation was done via the linux tool `tc` and experiments were done to determine the proper `tc` settings. The following sections examines the bandwidth overhead of DASH.

3.2 Overhead of DASH

In the DASH MPD file, each quality level is defined by an average bitrate value listed in bits per second. DASH clients compare their bandwidth measurements to the average bitrate value to decide if the network has the capacity to allow them to stream at that quality level. To verify if

the average bitrate values listed in the MPD files provided by [28] reflect the actual bandwidth required, a simple experiment was done with as single DASH client that was configured to request only segments that have an average bitrate of 222 Kbps listed in the DASH MPD file. The default experimental setup shown in Fig. 3.1 was used with the bandwidth limit set to the same value as the average bitrate in the MPD file. The client used a buffer size of 30 seconds and was configured to start video playback as soon as the buffer was full.

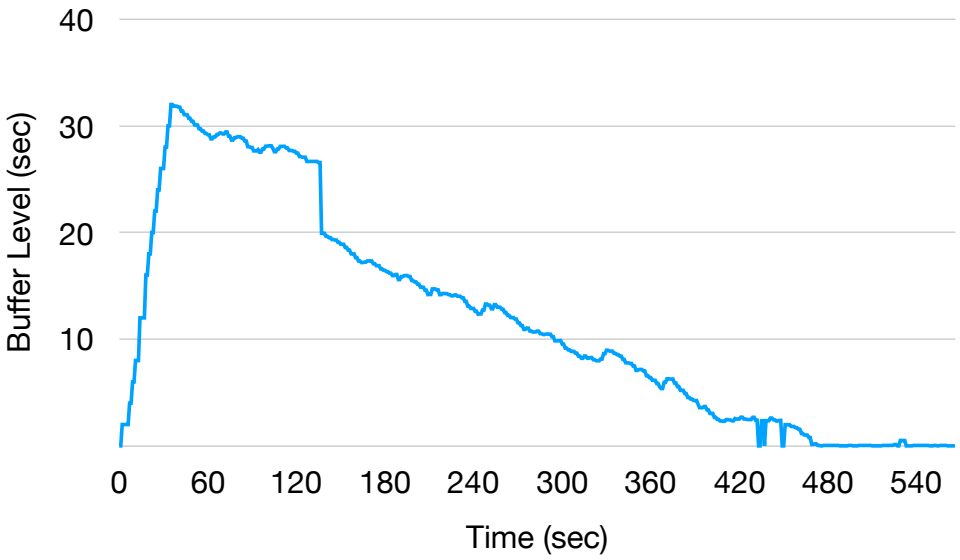

Figure 3.7: Buffer levels when a client requests only segments with a quality level of 222 Kbps and the bandwidth limit set to 222 Kbps.

As shown in Fig. 3.7, the buffer level had an overall downward trend resulting in stalls in video playback near the end of the video stream. The initial buffering time was 32.89 seconds, which means it took slightly over 30 seconds to download 30 seconds of video content. This suggests there is a small amount of overhead associated with DASH that needs to be considered if the results above are to be avoided. To determine the magnitude of the overhead, we found the minimum bandwidth for each quality level provided, using a harmonic mean window of 1, and then compared this to the bitrate listed in the MPD file to determine the overhead of DASH. For this experiment, a bandwidth supports a quality when at least 90% of the segments requested are of the

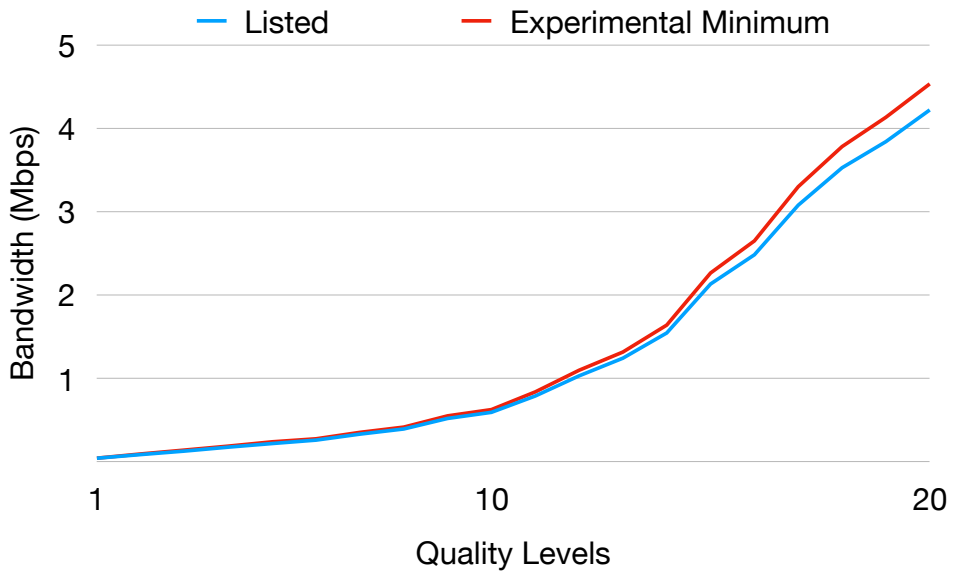

Figure 3.8: Listed bitrate in the MPD file vs experimentally found minimum bandwidth of quality levels for the “Big Buck Bunny” video.

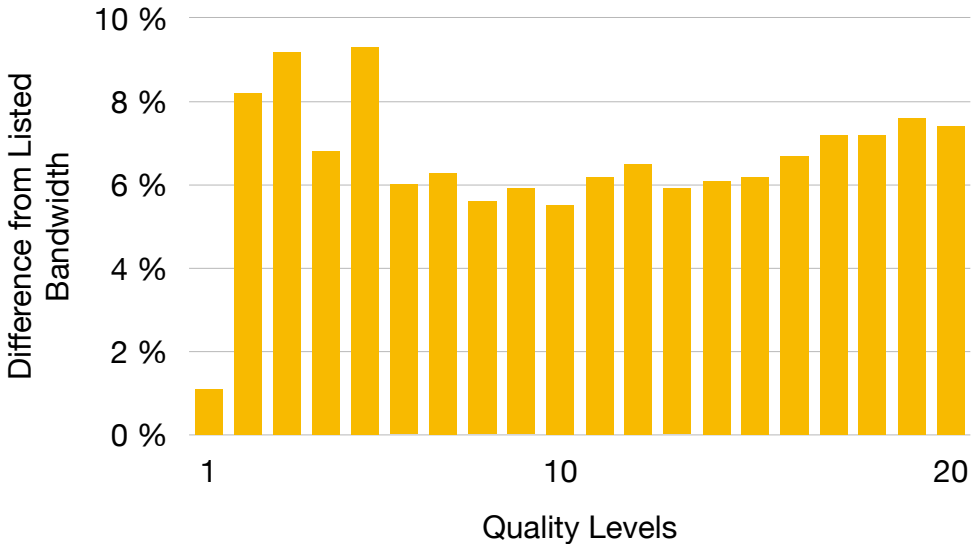

Figure 3.9: Percent difference between listed bitrate in the MPD file and the experimentally found bandwidth for each quality level of the video “Big Buck Bunny”.

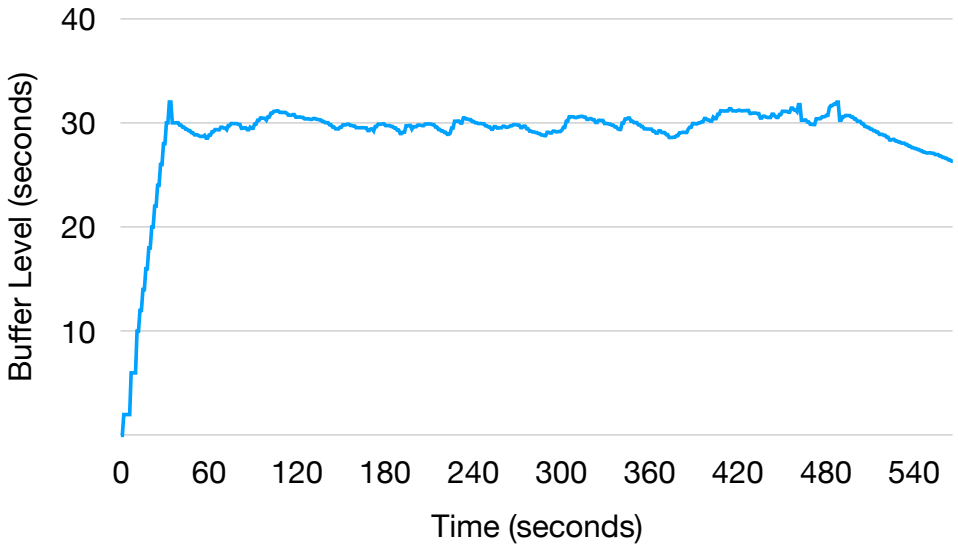

Figure 3.10: Buffer levels when a client requests only segments with a quality level of 222 Kbps and the bandwidth limit set to 236 Kbps.

quality level and are continuous. A harmonic mean window size of 1 is the most reactive to small network changes making it the least stable window size. If a bandwidth is deemed a minimum bandwidth for some quality with a harmonic mean window size of 1 then it is likely a minimum for a larger window size. The bitrate from the MPD file is shown as the blue line in Fig. 3.8 along with the average of 10 minimum bandwidth tests shown as a red line. The minimum bandwidth is 1.1% to 9.3% higher than the bitrate with an average of 6.5% and a median of 6.4%.

Next, we repeated the experiment with a client that requests segments only at the 222 Kbps level. However, this time we increase the bandwidth limit to $222 * 1.065 = 2.36$ Kbps to include the overhead. As shown in Fig. 3.10, the client is able to maintain a healthy buffer level throughout the entire streaming session, meaning that the session needs 236 Kbps bandwidth. We also noticed that the initial phase is now 30.8 seconds which is around 2 seconds less than the initial phase when the bandwidth limit was set to 222 Kbps. The healthy buffer levels coupled with the shorter initial buffering time show the need to consider overhead when choosing either quality levels or bandwidth limits.

There is a slight overhead associated with DASH that needs to be considered if clients are to have a high QoE. In this section, we conducted experiments to determine the minimum bandwidth for each quality level of the default video that were on average 6.5% higher than the average bitrate listed in the MPD file. When a client is choosing a quality level it would be helpful to consider this overhead in order to experience a high QoE. The following section compares the performance of different adaptive approaches in DASH.

3.3 Comparison of Adaptive Approaches

Chapter 2 categorized DASH adaptation algorithms into three broad categories, (1) rate-based algorithms, (2) buffer-based algorithms, and (3) hybrid algorithms. Rate-based algorithms utilize download rates of segments to estimate network conditions and request segments. Buffer-based algorithms utilize buffer levels to estimate network conditions and request segments. Hybrid algorithms utilize both download rates and buffer levels to estimate network conditions and request segments. This section will test a rate-based algorithm, a buffer-based algorithm, and a hybrid algorithm under three different network scenarios and compare the performance of each.

The rate-based algorithm used is simply the harmonic mean of the previous 20 segment download rates as a bandwidth estimation as done in [21]. The buffer-based algorithm is a server side algorithm based on estimated client buffer levels as presented in [23] and shown in Chapter 2. The hybrid algorithm is the default algorithm present in the GPAC player [27] and shown in Chapter 2. Each algorithm uses a buffer level of 30 seconds and does not begin playback until the buffer has been filled. The buffer-based algorithm is implemented on a proxy between the clients and the server that intercepts and modified segment requests while the rate and hybrid algorithms are implemented in the GPAC player.

Each algorithm will be tested under three network scenarios, the Static, Drop, and Short Drop network scenarios. To measure the performance of these algorithms, we will utilize the average number of quality switches for each client, the average buffer level for each clients, the bandwidth

utilization, and the average of the standard deviation between client quality levels. Each test will be repeated 10 times with 4 clients streaming simultaneously and the results will be averaged over all 10 runs. For each network scenario, there will be a table presenting the results for all clients and window sizes. The best results for each client will be in bold and if there is a tie each value will be in bold.

The **Static network scenario** consists of a constant bandwidth limit of 2.7 Mbps for 3 minutes which is sufficient for each of the 4 clients to stream at a quality level of 595 Kbps accounting for overhead. This network scenario will help to verify that each window size is able to accurately estimate a constant bandwidth.

	Fairness	**Util. (%)**	**Switches**	**Buffer (s)**	**Quality (Kbps)**
Buffer	147535.3	**92.2**	39.7	**26.6**	**622.5**
Rate	**133004.3**	91.8	**9.5**	26.4	619.7
Hybrid	142993.7	84.0	12.3	23.2	566.8

Table 3.3: Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the static network scenario.

The buffer-based algorithm is very unstable due to download overlap pointed out in [2] and reactions to small changes in buffer levels. This leads to very high number of quality switches and a low degree of fairness however, it also allows clients to utilize any unused bandwidth or switch to lower qualities with the small drop in buffer level. The rate-based algorithm produced the lowest number of quality switches due to its harmonic mean window size of 20 that is less reactive to small differences in throughput values. The static bandwidth setting allowed clients to accurately estimate bandwidth and settle into similar and stable quality levels resulting in the highest degree of fairness.

The low utilization, quality levels, and buffer levels of the hybrid algorithm shown in Table 3.3 are a result of its low and high buffer thresholds. Referring back to pseudo code for the hybrid algorithm in the related work Chapter, we can see that clients can only switch to a higher quality one level at a time and only if they have a high buffer level. Using the default high and low buffer thresholds of $b-l$ and l where b is the maximum buffer level of 30 seconds and l is the segment

length of 2 seconds, we get a high buffer threshold of 28 seconds and a low buffer threshold of 2 seconds. This means high buffer clients aggressively switch to higher quality levels and get stuck making it harder for other clients to switch to higher quality levels.

The **Drop network scenario** consists of a bandwidth limit of 2.7 Mpbs for 1 minute, 1.35 Mbps for 1 minute, and 2.7 Mbps for the final minute as shown in Fig. 3.11. This bandwidth limit will allow each of the 4 clients to stream at a quality level of 263 Kbps during the drop and 595 Kbps otherwise accounting for overhead. The drop network scenario will help us determine the ability of an algorithm to react to dynamic network conditions.

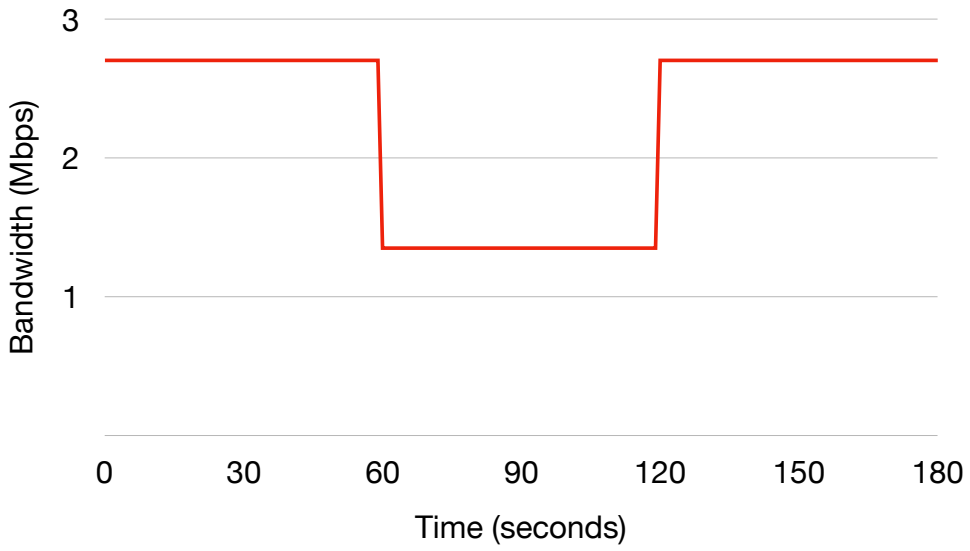

Figure 3.11: The drop network scenario.

	Fairness	**Util. (%)**	**Switches**	**Buffer (s)**	**Quality (Kbps)**
Buffer	134696.5	91.9	48.9	**26.3**	516.8
Rate	140914.5	**93.4**	**14.3**	25.7	**525.2**
Hybrid	**131833.8**	84.8	14.7	21.9	477.2

Table 3.4: Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the drop network scenario.

Similarly to the static scenario, the buffer-based algorithm produced the highest number of switches and the highest buffer levels due to its tendency to react to small changes in buffer levels.

However, it did not produce the highest utilization and quality levels. This is because the rate-based algorithm is less reactive to changes in bandwidth resulting in clients not immediately reacting to the drop in bandwidth. This lead to the rate-based algorithm having high utilization, quality levels, and low buffer levels.

As explained earlier, the hybrid algorithm leads to unfairness by allowing high buffer clients to use more than their fair share of bandwidth. The significant bandwidth drop present in this test along with the low buffer threshold prevented this from happening. Referring back to pseudo code for the hybrid algorithm in the related work chapter, we can see that clients can only switch to a lower quality level when they have a low buffer level. Using the default values of 30 second buffer and 2 second segments we get a low buffer level of 2 seconds. This means it is unlikely clients switched to lower quality levels during the bandwidth drop leading to low buffer levels. The low buffer levels eliminated the unfairness caused by high buffer clients getting stuck in high quality levels.

The **Short Drop network scenario** consists of a bandwidth limit of 2.7 Mbps for 3 minutes with a 2 second drop to 1.35 Mbps 28 seconds into each minute as shown in Fig. 3.12. This bandwidth limit will allow each of the 4 clients to stream at a quality level of 263 Kbps during the drops and 595 Kbps otherwise accounting for overhead. It is beneficial to have an algorithm that reacts quickly to bandwidth changes however, very short drops in bandwidth can be absorbed by the buffer without losses of QoE. Thus it is possible for an algorithm to be too reactive. The short drop scenario will help us determine if an algorithm is too reactive resulting in a loss of QoE.

	Fairness	**Util. (%)**	**Switches**	**Buffer (s)**	**Quality (Kbps)**
Buffer	155231.5	92.1	40.9	**26.7**	611.5
Rate	121721.9	**93.2**	**9.1**	26.4	**618.8**
Hybrid	**103149.0**	85.9	11.0	22.6	570.1

Table 3.5: Comparing the results of the buffer-based, rate-based, and hybrid algorithms under the short drop network scenario.

The results of the short drop test are very similar to the drop test and same explanations apply. The buffer-based algorithm produced a higher number of switches and high buffer levels due to its

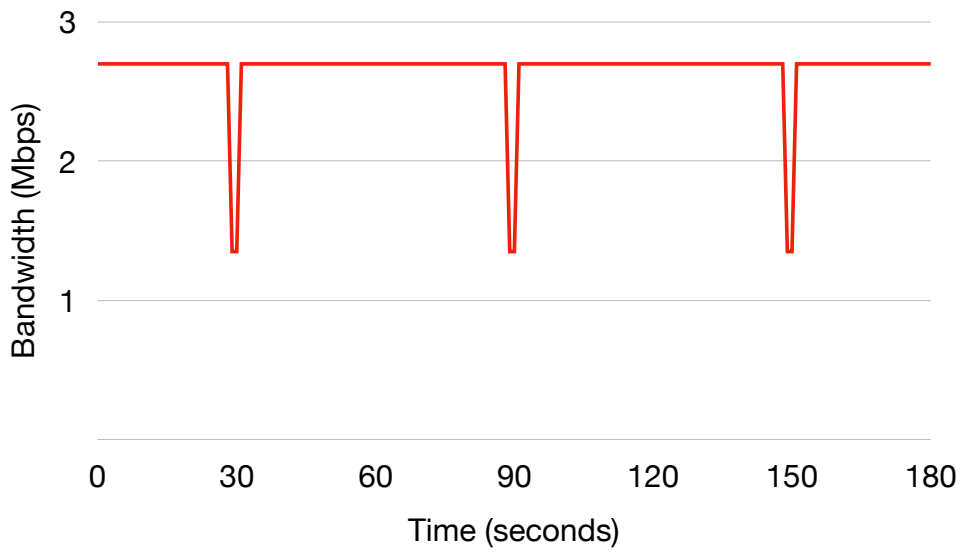

Figure 3.12: Short drop network scenario.

tendency to react to small changes in buffer levels. The rate-based algorithm produced the highest utilization and quality levels due to its tendency to not react to momentary changes in network conditions. The low buffer levels of the hybrid algorithm are caused by the inability of the hybrid algorithm to react to network changes which eliminated the unfairness caused by high buffer clients getting stuck in high quality levels.

These preliminary results show that each algorithm reacts differently to different network scenarios with the rate-based algorithm producing high utilization and low number of switches, the buffer-based algorithm producing high buffer levels, and the hybrid algorithm producing fairness. To draw any deeper conclusions we will need to perform more tests with a greater variety of network scenarios. The following section will perform these same tests but only use the rate-based algorithm with different window sizes to determine the effect window size has on algorithm performance.

3.4 Harmonic Mean Window Tests

In the previous section, we determined that the rate-based algorithm utilizing the harmonic mean of the previous 20 segments download rates (as done in [21]) performed well under dynamic network scenarios. Jiang *et al*. did not provide a justification for their choice of a window size of 20 in [21]. To determine the effect of window size of algorithm performance we will test window size of 10, 20, and 30 using the same scenarios as the previous section.

Overall, we predict that a window of 10 will result in the highest number of quality switches due to the window size being more susceptible to momentary fluctuations in throughput. We also predict that a window of 10 will result in the lowest degree of fairness due to the higher number of quality switches. Conversely, we predict that a window of 30 will result in the lowest number of quality switches due to the window size being less susceptible to momentary fluctuations in throughput. We also predict that a window of 30 will result in the highest degree of fairness due to the low number of quality switches experienced by each client. We also predict that each window size will perform similarly in terms of bandwidth utilization due to the fact that each window sizes will be using almost identical throughput values to estimate the bandwidth.

The **Static network scenario** consists of a constant bandwidth limit of 2.7 Mbps for 3 minutes which is sufficient for each of the 4 clients to stream at a quality level of 595 Kbps accounting for overhead. As expected, the window size of 30 produces the lowest number of switches, and highest degree of fairness. All other metrics are very similar between the three window sizes.

	Fairness	**Util. (%)**	**Switches**	**Buffer (s)**	**Quality (Kbps)**
Win of 10	180203.8	**93.2**	14.8	**26.6**	**629.2**
Win of 20	149277.5	**93.2**	9.0	26.4	628.8
Win of 30	**134255.9**	93.1	**7.5**	26.5	628.4

Table 3.6: Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the static network scenario.

The **Drop network scenario** consists of a bandwidth limit of 2.7 Mpbs for 1 minute, 1.35 Mbps for 1 minute, and 2.7 Mbps for the final minute as shown. This bandwidth limit will allow

each of the 4 clients to stream at a quality level of 263 Kbps during the drop and 595 Kbps otherwise accounting for overhead. The drop network scenario will help us determine the ability of a harmonic mean window size to react to drops in bandwidth. We predict that a window of 10 will result in the highest buffer levels due to its ability to react quickly to the bandwidth drop. We also predict that a window of 30 will result in the lowest buffer levels due to its inability to react quickly to the bandwidth drop.

	Fairness	**Util. (%)**	**Switches**	**Buffer (s)**	**Quality (Kbps)**
Win of 10	149733.2	93.3	19.9	**25.2**	524.8
Win of 20	**137521.4**	**94.7**	13.2	23.9	**532.6**
Win of 30	147698.7	93.8	**10.9**	23.3	527.6

Table 3.7: Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the drop network scenario.

The window size of 20 was able to produce the highest degree of fairness, highest utilization, and highest quality levels due to its compromise between stability and reactivity. When a client gathers a large number of high throughput values, due to them connecting first or getting lucky and frequently downloading segments alone, they can get stuck in a high quality level. As shown in Figures 3.14 and 3.13, client 1 ends up using much more than their fair share of bandwidth under a window size of 30 compared to a window size of 20. The window size of 10 also produced unfairness due to its unstable quality levels.

The **Short Drop network scenario** consists of a bandwidth limit of 2.7 Mbps for 3 minutes with a 2 second drop to 1.35 Mbps 28 seconds into each minute. This bandwidth limit will allow each of the 4 clients to stream at a quality level of 263 Kbps during the drops and 595 Kbps otherwise accounting for overhead. It is beneficial to have a window size that reacts quickly to bandwidth changes however, very short drops in bandwidth can be absorbed the buffer without losses of QoE. Thus it is possible for a window size to be too reactive, such as a window size of 1. The short drop scenario will help us determine if a window size is too reactive resulting in a loss of QoE. We predict that a window size of 10 will be too reactive resulting in a loss of QoE that could be avoided if the buffer were used to absorb the short bandwidth drops. We also predict that

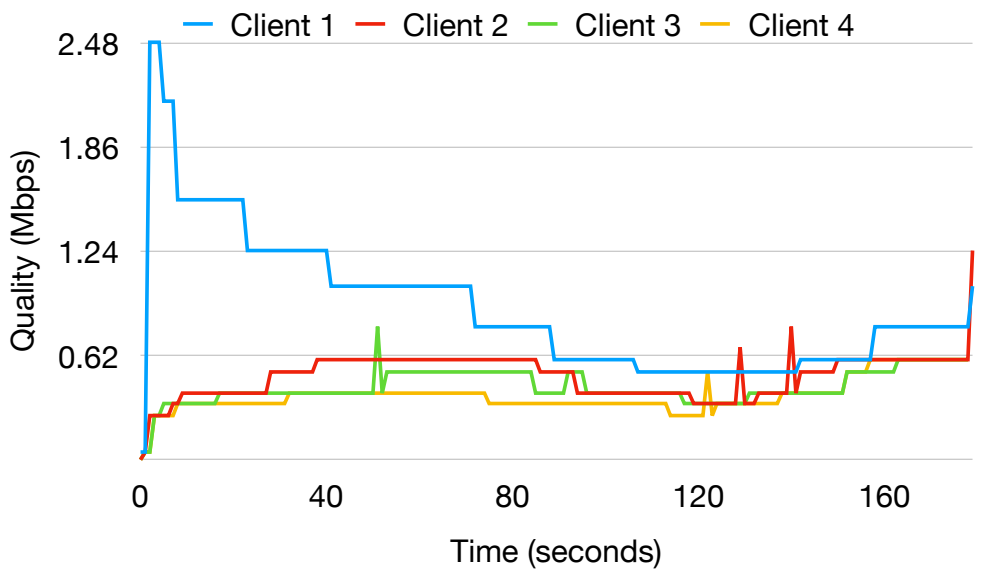

Figure 3.13: Quality levels of 4 clients over time using the harmonic mean of the previous 30 throughput values under the drop network scenario.

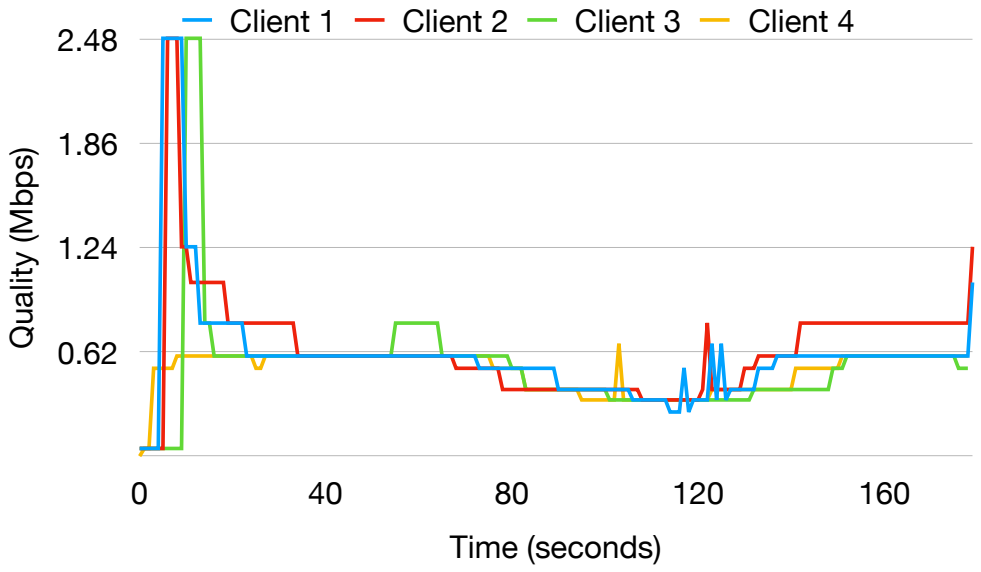

Figure 3.14: Quality levels of 4 clients over time using the harmonic mean of the previous 20 throughput values under the drop network scenario.

a window of 30 will not be too reactive and will allow the buffer to absorb the short bandwidth drops while maintaining a stable quality level.

	Fairness	Util. (%)	Switches	Buffer (s)	Quality (Kbps)
Win of 10	170725.8	92.6	15.5	26.4	614.6
Win of 20	**108219.2**	**94.2**	8.5	**26.5**	**625.6**
Win of 30	143921.2	91.6	**7.9**	26.1	607.7

Table 3.8: Comparing the results of the the rate-based algorithm using a harmonic mean window size of 10, 20, and 30 under the short drop network scenario.

The results of the short drop test are very similar to the drop test and the same explanations apply. The window size of 20 was able to produce the highest degree of fairness, highest utilization, and highest quality levels due to its compromise between stability and reactivity. This can be seen in Figures 3.16 and 3.15 where there is a more fair distribution of quality levels when using a window size of 20 compared to a window size of 30. The window size of 10 resulted in a low degree of fairness due to its instability.

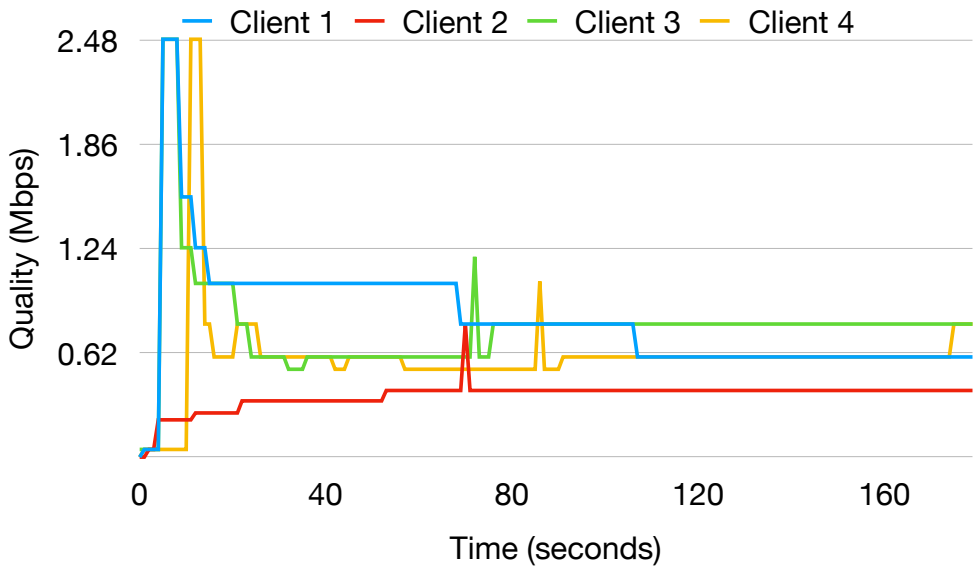

Figure 3.15: Quality levels of 4 clients over time using the harmonic mean of the previous 30 throughput values under the short drop network scenario.

The results of the harmonic mean window tests show that the harmonic mean window

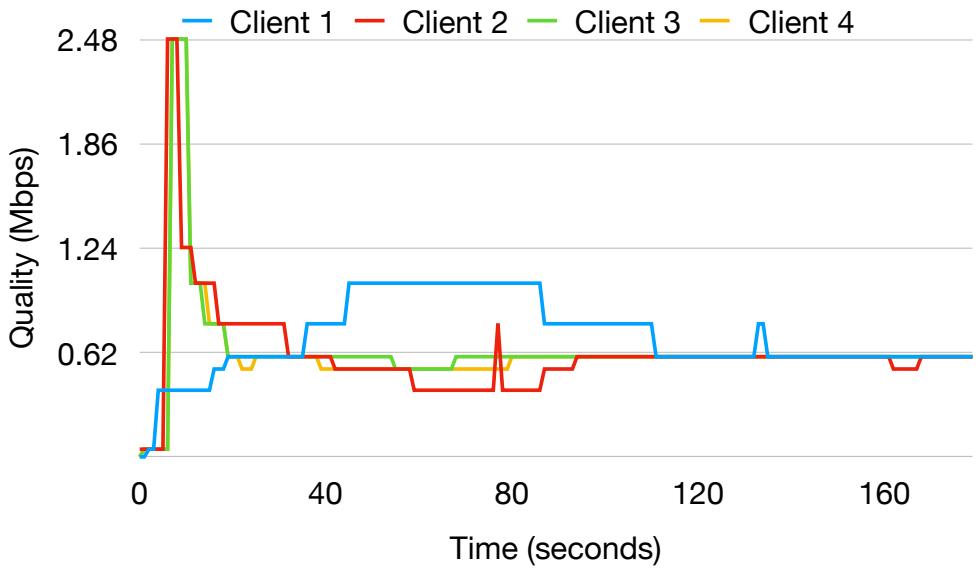

Figure 3.16: Quality levels of 4 clients over time using the harmonic mean of the previous 20 throughput values under the short drop network scenario.

size has a large impact on the visual stability of a streaming session with a window of 20 performing the best under dynamic network scenarios. This section tested harmonic mean window sizes of 10, 20, and 30 under static, drop, and short drop network scenarios. Each combination of window size and network scenario was tested with 4 clients streaming simultaneously. The drop and short drop network scenarios revealed that a window size of 30 can lead to unfair outcomes that take a long time to correct themselves due to the large window size. This section and the previous section showed the effectiveness of a window size of 20 with respect to dynamic network scenarios. The following section examines the effect that different startup strategies have on QoE of DASH clients.

3.5 Initial Phase

If a client is utilizing a harmonic mean of the last n throughput values as a bandwidth estimation, then that client may experience a long and unstable initial phase if n is sufficiently large. For example, if $n = 20$ and the client has only downloaded 5 segments then the harmonic mean window

size if effectively 5 since the client only has 5 throughput values available. This default behaviour increase the chances of a quality switch during the initial phase due to the unstable window sizes being used. Additionally, if the client is only using a few throughput values to estimate bandwidth it may over estimate leading to a very long initial phase. Mok *et al.* suggest that long initial phase lengths are undesirable due to the fact that they contribute negatively to a users QoE [37]. Rodriguez *et al.* suggest that unstable initial phases are undesirable because quality switches near the beginning of a streaming session have a greater negative affect on users QoE than quality switches near the end of a streaming session [47].

To shorten and stabilize the initial phase, there are two general approaches: (1) lowest quality strategy - request the lowest quality during the initial phase to minimize the length of the initial phase and best stability, and (2) midway quality strategy - request a quality level that is half of the clients fair share of the servers bandwidth during the initial phase. This will provide better visual quality while keeping the initial phase short and stable. To test these two strategies, an experiment was done using the default experimental setup with two clients and a bandwidth limit of 2.64 Mbps. This bandwidth limit should allow each client to stream at a quality level of 1.24 Mbps accounting for overhead. Each client used a buffer level of 30 seconds and was configured to start video play back when the buffer was full. We will examine the buffer levels and segment qualities for each strategy and compare them to the default behaviour. By default, GPAC is configured to request the lowest quality for the very first segment which is what a client using default behaviour will do.

As shown in Fig. 3.17, the default behaviour resulted in the longest initial phases at 36 seconds for client 1 and 38 seconds for client 2. In contrast to this, the lowest quality strategy resulted in the shortest initial phases of 1 second for client 1 and 1.3 seconds for client 2 as shown in Fig. 3.18. Finally, using the midway quality strategy resulted in initial phase lengths of 16.2 seconds for client 1 and 13.7 seconds for client 2 as shown in Fig. 3.19. The very short initial phase lengths provided by the lowest quality strategy would improve the QoE of users however, this benefit may be undermined by the low video quality during the initial phase and the low buffer levels show in

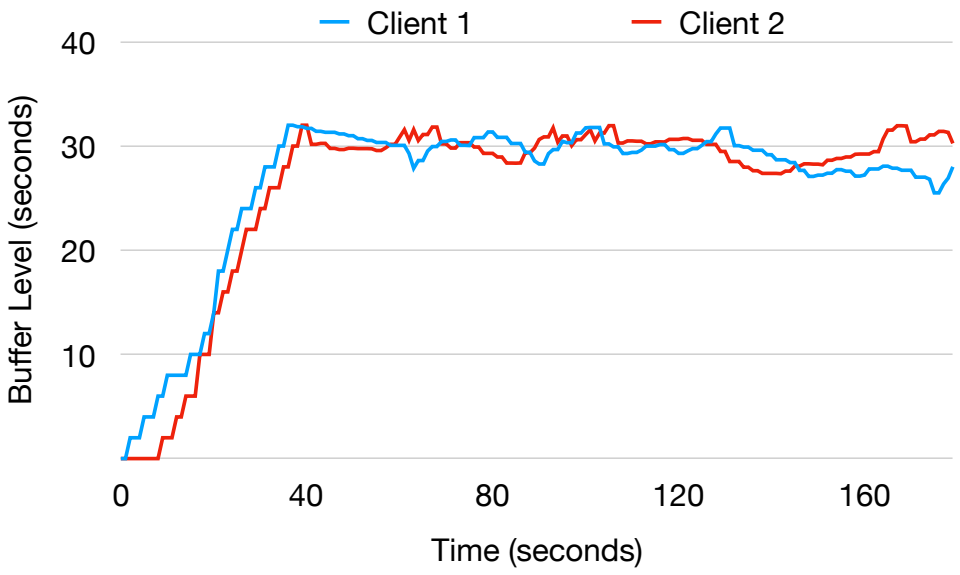

Figure 3.17: Buffer levels with a harmonic mean window of 20 using the default behaviour.

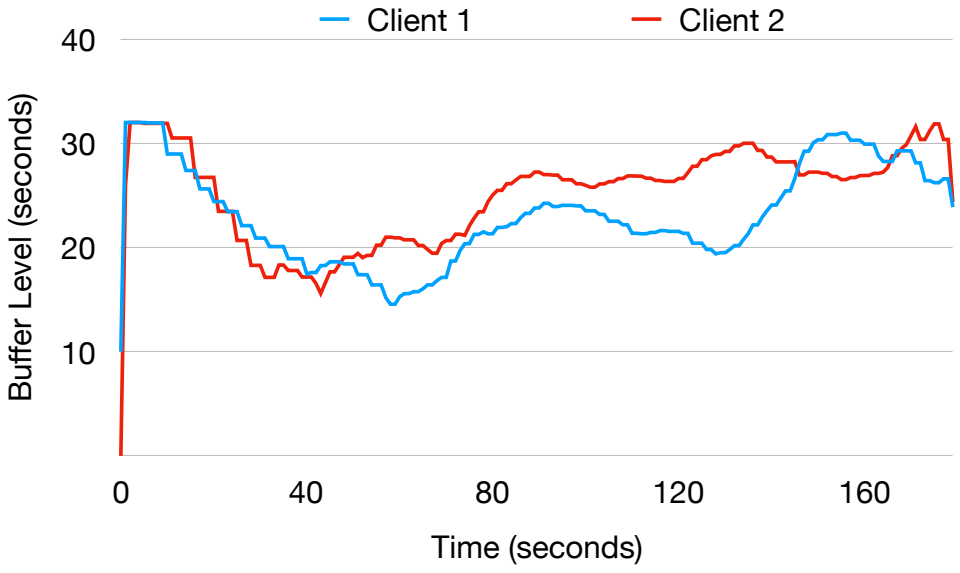

Figure 3.18: Buffer levels with a harmonic mean window of 20 using the lowest quality strategy.

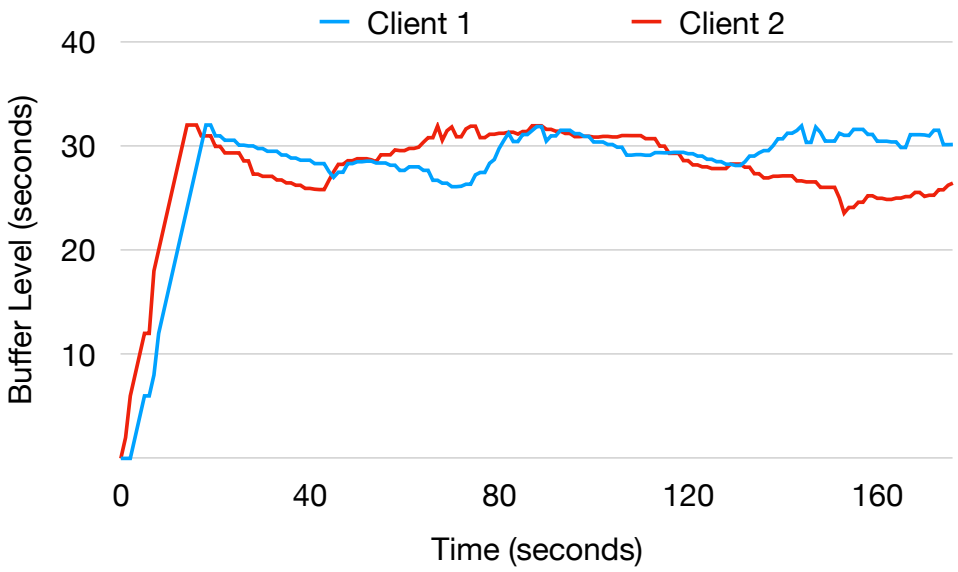

Figure 3.19: Buffer levels with a harmonic mean window of 20 using the midway quality strategy.

Fig. 3.18. Next we will examine the quality levels during each test that will help explain the low buffer levels that resulted from using the lowest quality strategy.

As shown by Fig. 3.20, the default strategy results in an initial phase that is not visually stable due to the harmonic mean window being effectively equal to the number of segments downloaded. When using the default strategy, each client begins by downloading the lowest quality possible for the first segment and then uses the throughput value from the first segment only to estimate the bandwidth. As a result, there is a large jump in quality level between the first and the second segments.

There is also a large spike in quality levels directly after the initial phase ends when clients use the lowest quality strategy as shown in Fig. 3.21. This is the result of clients overestimating the bandwidth due to high throughput values experienced during the initial phase. These high throughput values during the initial phase are due to the small sizes of video segments at the lowest quality level. The small segment sizes allow clients to download the segments very quickly which leads to less time spent occupying the link between the client and the server. This ultimately leads to a higher chance of each client occupying the link alone when downloading segments which leads

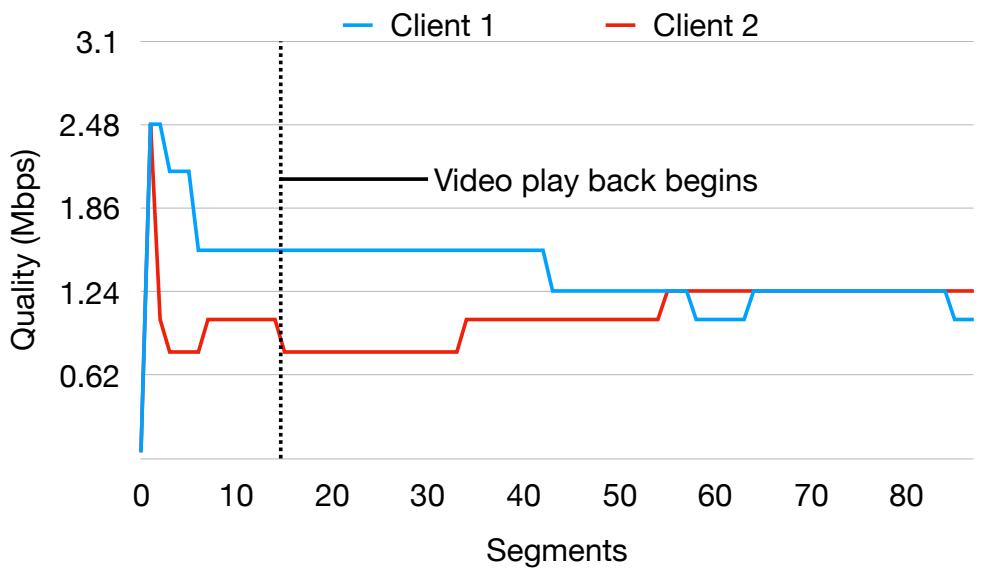

Figure 3.20: Quality levels with a harmonic mean window of 20 using the default behaviour.

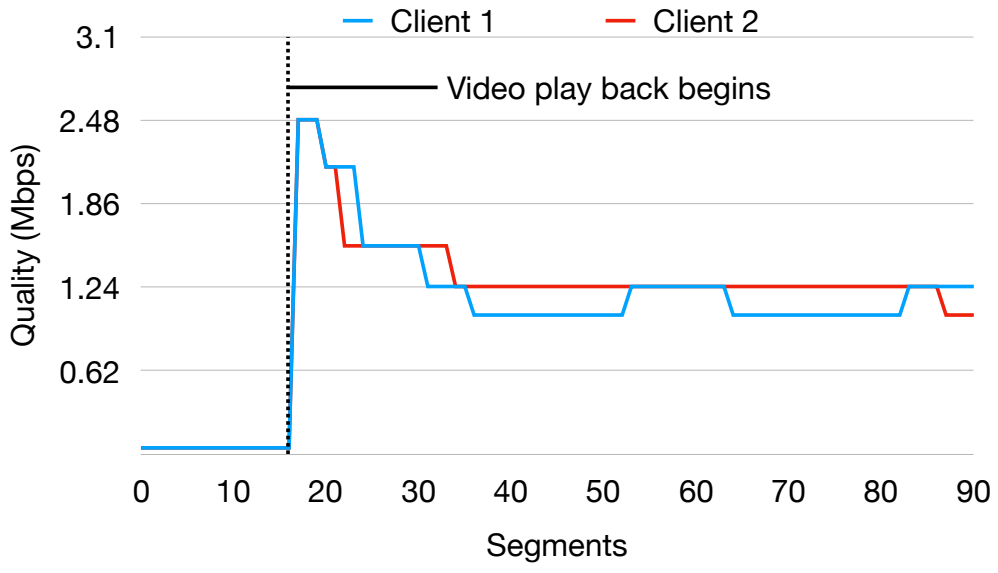

Figure 3.21: Quality levels with a harmonic mean window of 20 when using the lowest quality strategy.

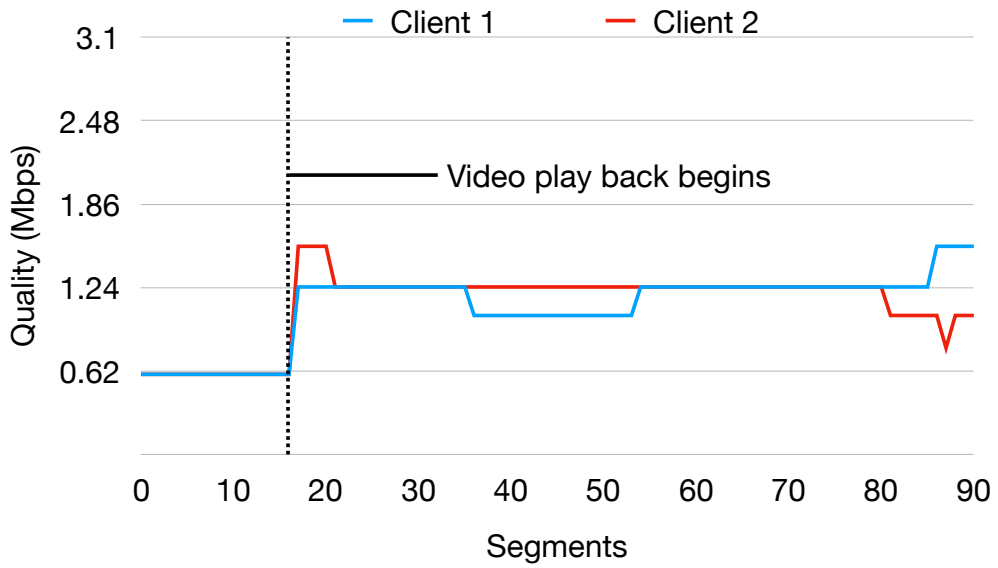

Figure 3.22: Quality levels with a harmonic mean window of 20 when using the midway quality strategy.

to high throughput values. Using the midway quality strategy eliminates this issue and allows each client to obtain throughput values that more accurately reflect network conditions as shown in Fig. 3.22.

Configuring DASH clients to request midway quality segments during the initial phase results in a stable initial phase that allows clients to gather throughput values that accurately reflect network conditions. In this section, we showed that clients using a harmonic mean of the last 20 throughput values as bandwidth estimation may experience long and unstable initial phases. Mok *et al.* and Rodriguez *et al.* suggest that long and unstable initial phases negatively affect users QoE [37, 47]. To address this, we deployed the lowest quality strategy and the midway quality strategy. The lowest quality strategy resulted in a very short and stable initial phase however, it also resulted in very low video quality during the initial phase and lead to clients over estimating network conditions. The midway quality strategy resulted in a stable initial phase that was longer than the lowest quality video strategy but short than the default behaviour. The video quality is much higher during the initial phase than the lowest quality strategy and it also lead clients to more

accurately estimate network conditions. One drawback of this approach is that it needs global information that is only available on the server side. The following section examines the need for server side intervention to address fairness issues that arise when multiple clients are streaming simultaneously.

3.6 Fairness Across Sessions

As shown in [2], fairness issues can arise when two or more clients are streaming simultaneously. This is the result of either (1) a group of two or more clients often downloading segments at the same time leading them to under estimate network conditions or (2) clients often downloading segments alone, leading them to over estimate network conditions. To demonstrate this, the following test uses the default setting with a bandwidth limit of 2.64 Mbps and two clients that are both using the harmonic mean of the last 20 throughput values. The bandwidth limit of 2.64 Mbps is sufficient for both clients to stream at a quality level of 1.24 Mbps accounting for overhead.

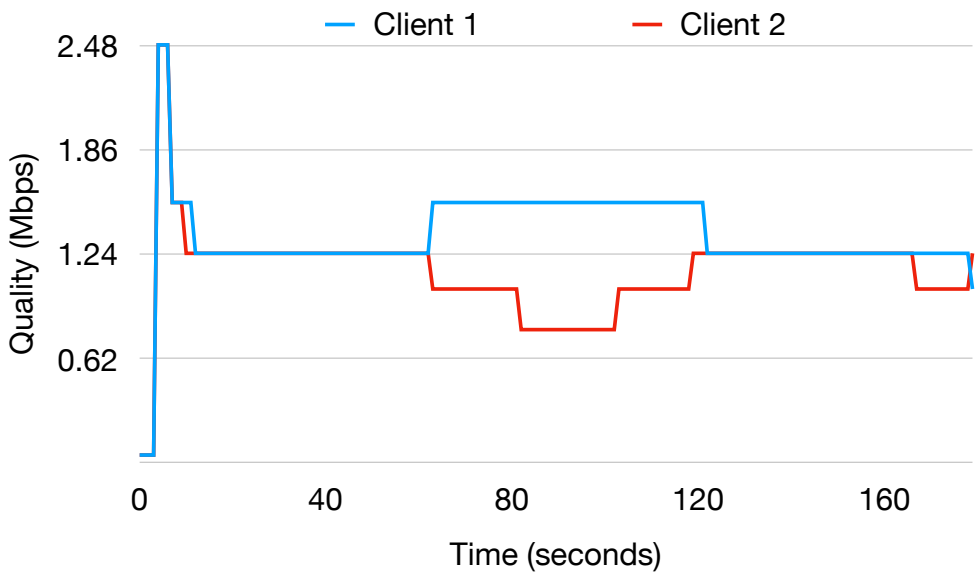

Figure 3.23: Quality levels of 2 clients using the harmonic mean of the previous 20 throughput values with a bandwidth limit of 2.64 Mbps.

As shown in Fig. 3.23, client 1 ends up getting a greater share of the network bandwidth than

client 2. Jiang *et al.* proposed an adaptation algorithm to address this issue called FESTIVE that is presented in chapter 2 [21]. FESTIVE adds a small random delay before each clients request to reduce the chances of these clients getting stuck in a cycle of downloading segments together or never downloading segments together. FESTIVE also utilizes a ramp up mechanism that makes it harder for higher quality clients to switch to an even higher quality level preventing them from utilizing more than their fair share of bandwidth. The same test was repeated using the FESTIVE algorithm.

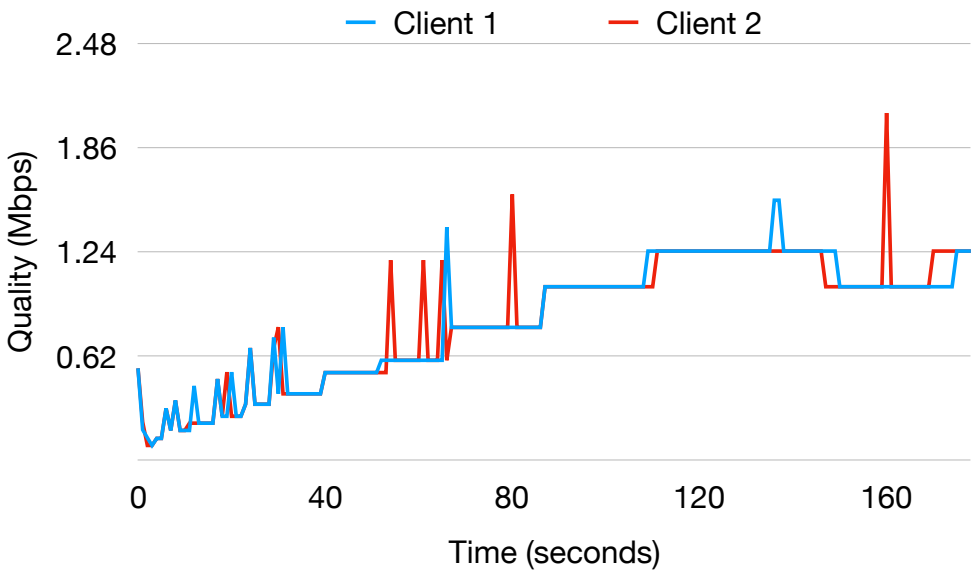

Figure 3.24: Quality levels of 2 clients using the FESTIVE algorithm with a bandwidth limit of 2.64 Mbps.

As shown in Fig. 3.24, FESTIVE results in nearly identical quality levels for client 1 and client resulting in a high degree of fairness (the large spikes in quality level are due to multiple requests with a second and not momentary jumps in quality). FESTIVE is a client side solution to the fairness issues presented in [2], we can also utilize the server to address these issues. If we utilize the proxy setting, where clients requests are first tunneled through a proxy before being sent to the server, the proxy can analyze clients requests and modify them based on the server bandwidth. The proxy can give each client a fair share of the servers current bandwidth, accounting for overhead.

During the initial phase the proxy simply sends each client the lowest quality level available. The same test was repeated using the server side solution described.

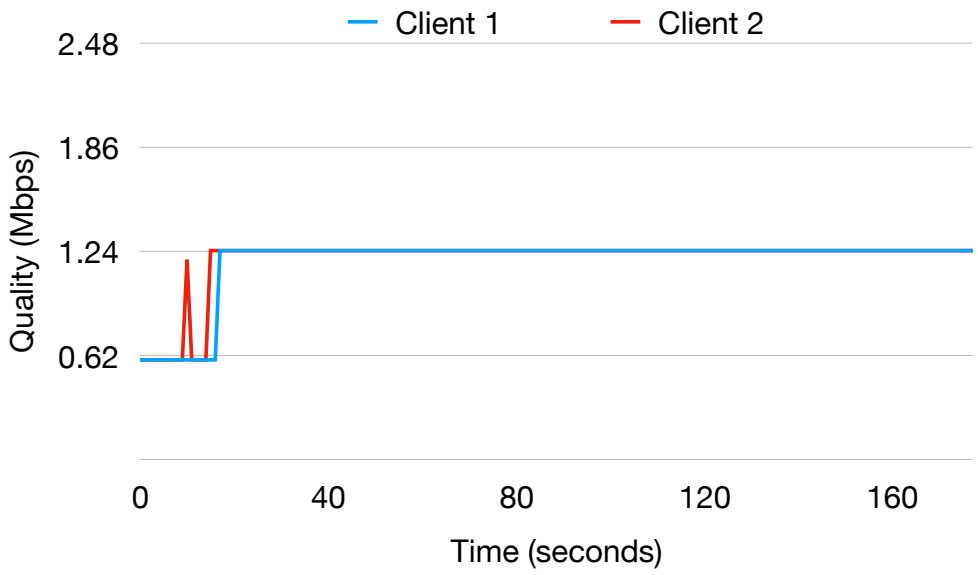

Figure 3.25: Quality levels with a bandwidth limit of 2.64 Mbps and a proxy modifying client requests.

As shown in Fig. 3.25, each client is given an equal share of the servers bandwidth resulting in both clients streaming at the same quality level and a high degree of fairness. For this test each client has the same bandwidth requirements thus giving them an equal share of bandwidth results in fairness. If clients were to have different bandwidth requirements (such as 4k television and a smartphone) then FESTIVE, similar algorithms such as [13] and [7], and the server side solution presented would result in resource fairness (equal distribution of bandwidth) but not necessarily QoE fairness. Researchers have considered these scenarios and created algorithms that attempt to achieve QoE fairness as opposed to resource fairness [10, 16]. These algorithms both utilize the Structural Similarity Index (SSIM) as a measure of QoE and attempt to achieve fairness between the SSIM measure between clients.

SSIM is a reference based, subjective quality measure of images that is easily computed and correlates well with human perception [62]. SSIM can be used to measure video quality by averag-

ing SSIM values over all frames of a video as done in [10] and [16]. SSIM values range from 0 to 1 where a value of 1 means exact similarity (a video would a have a SSIM value of 1 when compared to itself) and a value of 0 means no similarity. As shown in Table 3.9, we have computed the SSIM values of the video Big Buck Bunny at three resolutions (360p, 720p, 1080p). The reference video for each resolution is the highest quality video offered at that resolution. To compute the SSIM value of a video that has a different resolution than the reference video we first upscale the video to the same resolution as the reference video and then compute a SSIM value. This technique is used in [10] and is also what a video player would do when a video segment is downloaded that is a different resolution than the clients screen. As shown in Table 3.9, a video with the same bitrate results in different SSIM values depending on client resolution. This means that providing clients with equal bandwidth (resource fairness) may not result in similar SSIM values (QoE fairness). As shown in Fig. 3.26, to achieve a SSIM value of 0.94 we would need a bitrate of around 500 Kbps for a 720p client and a bitrate of around 1 Mbps for a 1080p client.

		SSIM		
Quality	**Resolution**	**360p**	**720p**	**1080p**
178 Kbps	480x360	0.937617	0.874103	0.844944
222 Kbps	480x360	0.953646	0.894626	0.864652
263 Kbps	480x360	0.963134	0.908238	0.878324
334 Kbps	480x360	0.973549	0.924399	0.895398
396 Kbps	480x360	1	0.933925	0.906005
522 Kbps	854x480		0.943478	0.918096
595 Kbps	854x480		0.950922	0.926849
791 Kbps	1280x720		0.954965	0.93014
1 Mbps	1280x720		0.968168	0.946091
1.2 Mbps	1280x720		0.974747	0.95461
1.5 Mbps	1280x720		1	0.963415
2.1 Mbps	1920x1080			0.968729
2.5 Mbps	1920x1080			0.97408
3.1 Mbps	1920x1080			0.98036
3.5 Mbps	1920x1080			0.983855
3.8 Mbps	1920x1080			0.986035
4.2 Mbps	1920x1080			1

Table 3.9: SSIM values for the “Big Buck Bunny” video at 3 resolutions (360p, 720p, 1080p).

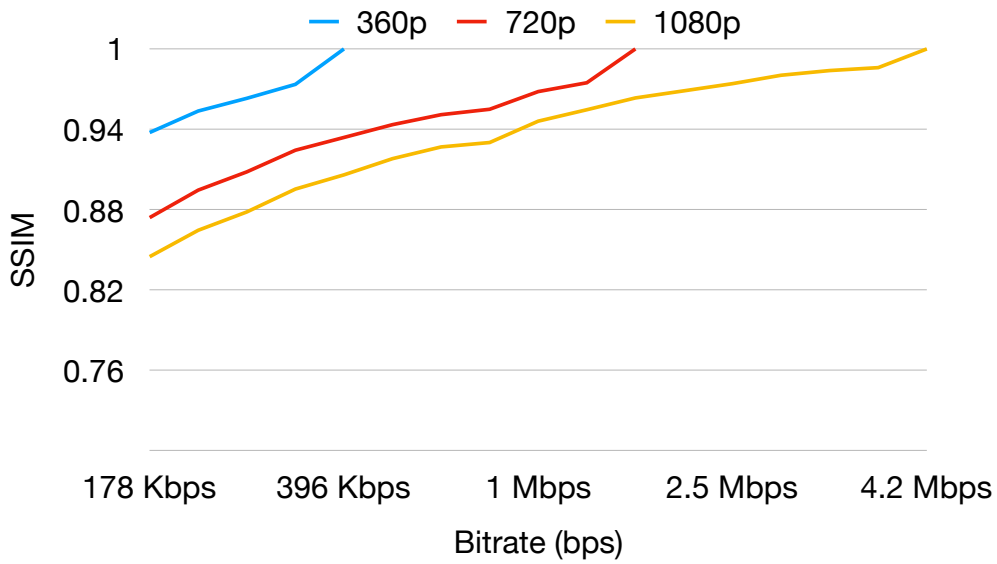

Figure 3.26: Comparison of SSIM values for the "Big Buck Bunny" video at 3 resolutions (360p, 720p, 1080p).

As shown in Table 3.9, if the previous tests were done with one 720p client and one 1080p client then providing them both with a quality of 1.2 Mbps would result in a SSIM of 0.974747 and 0.95461 for the 720p client and the 1080p client, respectively. This means the previous strategies would achieve resource fairness but not QoE fairness. To address this, Cofano *et al.* utilize an algorithm that attempts to maximize the lowest SSIM value over all clients as presented in Chapter 2. This algorithm first sets each client to the lowest quality level available. It then iteratively raises the quality level of the client with the lowest SSIM by one level until all the servers bandwidth has been used.

We will use a variation of the algorithm presented in [10]. Our algorithm utilizes both the server and the client and has clients use the harmonic mean of the previous 20 segment downloads as bandwidth estimation. The server then sends the client the minimum of the clients requests and the quality level determined by the algorithm presented in [10]. The same test will be repeated using the proxy setting and running the resolution aware algorithm presented with one 720p client and one 1080p, the results are shown below.

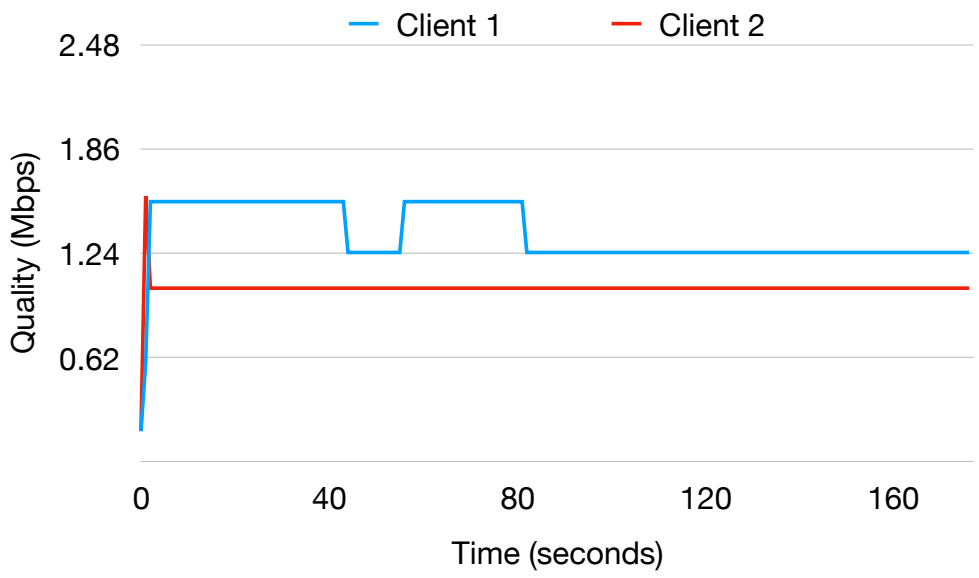

Figure 3.27: Quality levels of 2 clients using the resolution-based algorithm with a bandwidth limit of 2.64 Mbps.

As shown by Fig. 3.27, the 1080p client (client 1) streams at a mixture of quality levels consisting of 1.2 Mbps and 1.5 Mbps and the 720p client (client 2) streams at a quality level of 1 Mbps. This would result in SSIM values of roughly 0.95461 and 0.968168 for client 1 and client 2, respectively. This is in contrast to treating the clients equally resulting in a SSIM of 0.974747 and 0.95461 for the 720p client and the 1080p client respectively. Clearly this resolution aware approach is able to achieve a higher degree of QoE fairness than FESTIVE or the server side solution presented.

By utilizing global information (such as the servers bandwidth, the number of the clients currently streaming, and the resolutions of each client), a server side entity can assign quality levels to clients that result in a high degree of QoE fairness as opposed to resource fairness. In this section, we showed the fairness issues that arise when two clients are streaming simultaneously. To address this we utilized FESTIVE as presented in [21] and a server side entity to modify client requests. These solutions both provide a high degree of resource fairness, that is they allocate an equal share of bandwidth to each client currently streaming. This is ideal when clients

have the same bandwidth requirements however, this is less than ideal when clients have different bandwidth requirements. To achieve QoE fairness, an algorithm must provide an equal QoE to all clients which may not necessarily result in resource fairness. This was achieved using a server-side modification of the algorithm presented in [10] which is a based on SSIM and client resolutions.

3.7 Summary

In this chapter we performed a systematic analysis on the impact of fairness, bandwidth utilization,buffer level, visual stability, and visual quality on QoE under various controlled emulated network conditions. We began this chapter by outlining the performance metrics, such as segment quality, buffer levels, quality switches, initial phase length, and bandwidth utilization. It is worth noting that these metrics are largely independent of each other and it is standard practice to measure and compare them in isolation. For example, comparing the buffer levels produced by two different algorithms would be fair as long as the same maximum buffer size is used for both algorithms. We then presented our experimental setup up that allows us to repeat experiments using different adaptation schemes under various network conditions.

The main contributions from each section are as follows. Section 3.2 determined that on average a bandwidth 6.5% greater than the average bitrate listed in the MPD for a quality level is needed to steadily stream at that quality level. Section 3.3 tested a rate-based, buffer-based, and hybrid algorithm under different network conditions. It was determined that the rate-based algorithm produced high utilization and a low number of switches, the buffer-based algorithm produced high buffer levels, and the hybrid algorithm produced a high degree of fairness. Section 3.4 compared the performance of different version of the rate based algorithm and found that the harmonic mean window of 20 performed the best under dynamic network scenarios. Section 3.5 tested the lowest quality and midway quality strategies during the initial phase and found that the midway quality strategy results in a high degree of visual stability and allows clients to gather throughput values that accurately reflect network conditions. Section 3.6 presented different solutions to

fairness issues that arise when multiple clients compete for bandwidth. We found that algorithms can produce resource fairness via client-side and server-side solutions however, to produce QoE fairness a server-side solution is more feasible. The following Chapter will present a comparison of adaptation algorithms using network and client scenarios not using in this Chapter.

Chapter 4

Comparison of Adaptation Algorithms

In Chapter 3, we performed an analysis of QoE and fairness in DASH that utilized a rate-based, a buffer-based, and a hybrid algorithm. We tested each algorithm under one static and two simple dynamic network scenarios and found that the buffer-based algorithm produced the highest buffer levels, the hybrid algorithm produced good fairness, and the rate-based algorithm produced the highest utilization and quality levels. We then presented FESTIVE as a client-side solution to fairness issues and the resolution-based algorithm as a server-side solution to fairness issues. These preliminary results reveal some strengths and weaknesses of each algorithm however, more tests are needed to further explore each algorithms strengths and weaknesses.

In this Chapter, we will perform a cross comparison of adaptation algorithms with the goal of determining each algorithms strengths and weaknesses. The results of this comparison will inform the design of our new adaptation algorithm, FairQ. To do this, we will use more realistic network and client scenarios than the scenarios used in Chapter 3. Section 4.1 will begin by explaining our experimental setup used to test each algorithm. Section 4.2 will test each algorithm under a variety of bandwidth drops at different frequencies and magnitudes. Section 4.3 will test each algorithm using a mixture of bandwidth drops and raises. Section 4.4 will utilize a real bandwidth trace to further test the algorithms and section 4.5 will conclude the Chapter with a brief summary.

4.1 Experimental Setup

The algorithms used in this Chapter consist of (1) the harmonic mean of the last 20 throughput values as done in [21] (rate-based), (2) the buffer-based algorithm presented in [23] (buffer-based), (3) the default GPAC algorithm as a hybrid algorithm [27] (hybird), a resource fairness algorithm called FESTIVE [21], and a QoE fairness algorithm that is a modification of the algorithm pre-

sented in [10] (resolution-based). The network scenarios used to test these algorithms are all dynamic and will be presented along with the results. Each network scenario will be tested with 4 clients in both a homogeneous and mixed client scenario except for the final test which will use a bandwidth trace and 20 clients under a mixed client scenario. The homogeneous client scenario consists of exclusively clients with a resolution of 1920x1080 and the mixed client scenario consists of half clients with a resolution of 1920x1080 and the other half with a resolution of 480x360. For each algorithm, all client side adaptation is implemented in the adaptation module of the GPAC 0.6.1 player. All server side adaptation is implemented in the adaptation module of a python proxy we have developed. The setup is identical to the setup provided in Chapter 3 with the video server and the proxy each running their own virtual machine hosted by Cybera at the University of Calgary. Each machine has 8GB of ram, 4 virtual CPUs, and 40GB of storage with Ubuntu 14.04 LTS. All clients are running on the same machine that has 8GB of ram, 2 CPUs, 250GB of storage with Ubuntu14.04 LTS and video playback disabled to minimize CPU usage. The video used is "Big Buck Bunny" provided by Lederer *et al*. [28]. All bandwidth emulation is done via `tc`, a Linux program for configuring packet scheduling.

In the experiment setting for this Chapter, the computer running the clients is located at the same university as Cybera however, the connection between the clients and the server is through a real university network that could potentially impact test results. To determine if the network could be impacting test results, we first want to determine the capacity of this network by measuring the download rate of segment during a streaming session of a single client with no bandwidth limit. As shown in Fig. 4.1, the download rates range from 104 Mbps to 675 Mbps with an average download rate of 303 Mbps. The network scenarios we use in this Chapter have average bandwidth limits of 9 Mbps to 15 Mbps meaning we are using a small fraction of the total capacity of this network. Thus, we can be fairly confident that the network does not have a significant impact on test results.

Each experiment was repeated 3 times and the results were averaged over the 3 runs. We have assumed that clients cannot request video segments with a higher resolution that their screen

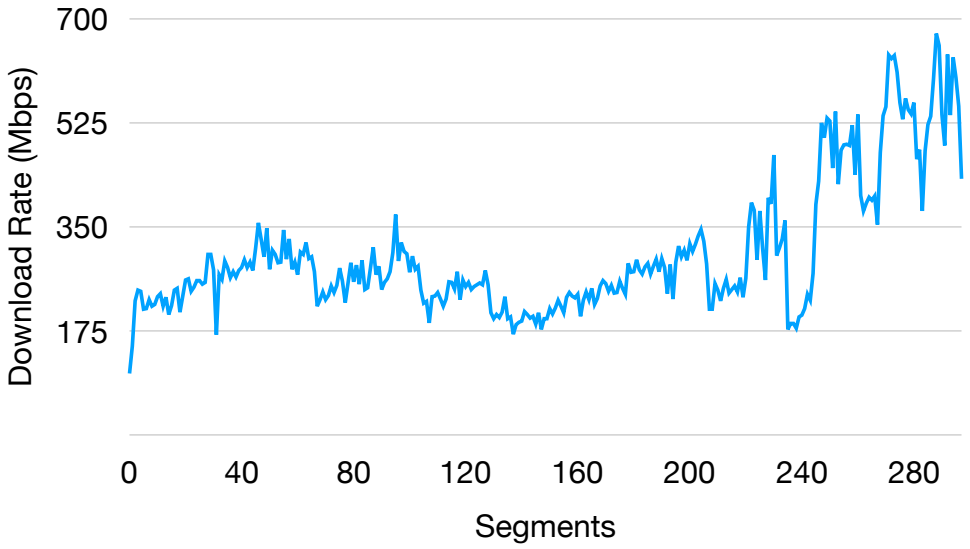

Figure 4.1: Download rate of segments when a client connects to the server with no bandwidth limit in place.

resolution. To achieve this we have simply created two versions of the MPD filed offered by the server, one that includes videos playable by a 1920x1080 client and another that only includes videos playable by a 480x360 client. The resolution based algorithm utilizes the bandwidth of the server when determining quality levels. The bandwidth of the server was set to the average of the corresponding network scenario when the tests were run.

For each scenario a table will be presented to summarize the results of that scenario. Each table will use the following metrics:

- **Fairness**: The standard deviation of average SSIM between clients over all runs. In this Chapter, we use mixed client scenarios (clients have different screen resolutions) thus, ensuring clients stream at the same quality level would not produce QoE fairness. To produce QoE fairness, we need to ensure clients stream at quality levels that result in equal QoE. Since we are using SSIM as an objective measure of QoE, the standard deviation of SSIM between clients is an appropriate measure of fairness. The standard deviation of QoE between clients is used as a fairness metric

in [40] and [41] and the variance in used in [58].

- **Utilization**: The average bandwidth utilization over all runs, listed as "Util." in the table.
- **Average Number of Quality Switches**: The average number of quality switches for all clients over all runs, listed as "Switches" in the table.
- **Average Buffer Levels**: The average buffer level for all clients over all runs in seconds, listed as "Buffer" in the table.
- **Initial Phase Length**: The average initial phase length for all clients over all runs in seconds, listed as "I.P. Length" in the table.
- **Average Segment Quality**: The average quality of segments for all clients over all runs in Mbps, listed as "Quality" in the table.

4.2 Bandwidth Drops

This section will test each algorithm under 4 different network scenarios, referred to as drop 1, drop 2, drop 3, and drop 4 as shown in Fig. 4.2. Drop 1 and drop 2 consist of bandwidth limits that fluctuate between 11 Mbps and 9.5 Mbps at frequencies of 30 seconds and 15 seconds respectively. A bandwidth limit of 11 Mbps should allow each client to stream at a quality of 2.5 Mbps and a bandwidth limit of 9.5 Mbps should allow each client to stream at a quality of 2.1 Mbps under homogeneous client scenarios. A bandwidth limit of 11 Mbps and 9.5 Mbps should allow both groups to stream at their maximum quality levels of 4.2 Mbps and 396 Kbps respectively under mixed client scenarios.

Drop 3 and drop 4 consist of bandwidth limits that fluctuate between 11 Mbps and 8 Mbps at frequencies of 30 seconds and 15 seconds respectively as shown in Fig. 4.2. A bandwidth limit of 11 Mbps should allow each client to stream at a quality of 2.5 Mbps and a bandwidth limit of

8 Mbps should allow each client to stream at a quality of 1.5 Mbps under homogeneous client scenarios. A bandwidth limit of 11 Mbps should allow both groups to stream at their maximum quality levels of 4.2 Mbps and 396 Kbps respectively under mixed client scenarios. A bandwidth limit of 8 Mbps should allow 1920x1080 clients to stream at a quality of 3.5 Mbps and 480x360 clients to stream at a quality level of 396 Kbps under mixed client scenarios.

We will present the results of the homogeneous client scenarios first followed by the mixed client scenarios. We will then re-analyze the results of the mixed client scenarios and take averages within the two resolutions groups and present the results of just 1080p clients and just 360p clients.

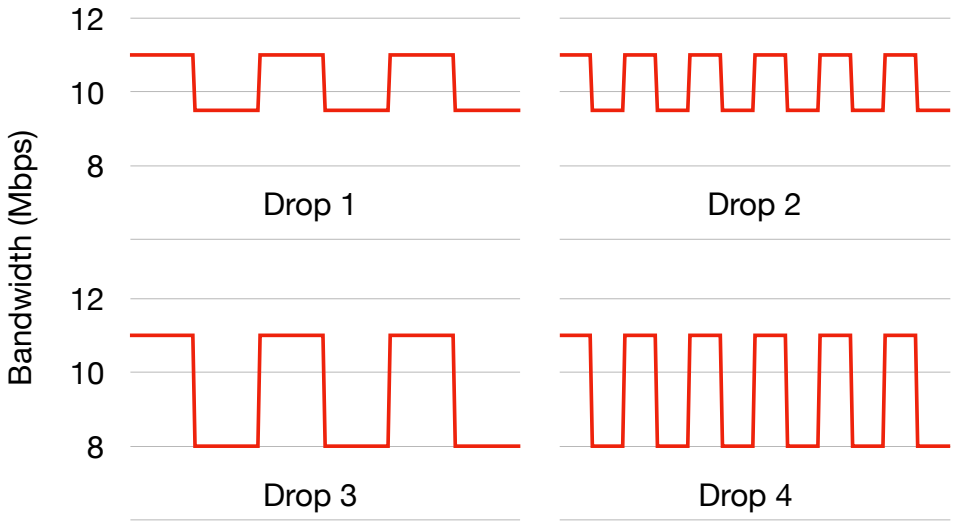

Figure 4.2: Drop 1 to 4 network scenarios.

4.2.1 Homogeneous Client Scenarios

Under the homogeneous client scenarios there are 5 trends that are present across all network scenarios. These trends can be seen in the results under the drop 1 network scenario shown in Table 4.1. The results in their entirety can be seen in Tables A.1 to A.4 in Appendix A. We will present these trends and explain why they persist across network scenarios.

Fairness: FESTIVE produced the best fairness and the hybrid algorithm produce the

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000853	78.3	**4.3**	27.3	**0.6**	2.056
Buffer	0.001680	86.8	24.0	25.6	16.6	2.278
Rate	0.001488	**91.9**	9.9	26.4	41.8	**2.411**
FESTIVE	**0.000438**	36.7	15.5	**29.0**	1.7	0.963
Hybrid	0.002561	53.3	15.8	23.8	**0.6**	1.400

Table 4.1: Results from the Drop 1 network scenario under a homogeneous client scenario.

worst fairness. This is due to the long ramp-up behavior of FESTIVE that results in client behaving almost identically for the majority of the stream as shown in Fig. 4.4. This behavior is largely unaffected by bandwidth changes due to the low utilization, thus it is present in every bandwidth scenario. The hybrid algorithm only allows high buffer clients to switch to higher quality levels and low buffer client to switch to lower quality levels. The hybrid algorithm uses a high buffer threshold of $b - l$ and a low buffer threshold of l where b is the maximum buffer length in seconds and l is the segment length in seconds. Using a buffer size of 30 seconds and a segment length of 2 seconds we get a high buffer threshold of 28 seconds and a low buffer threshold of 2 seconds. This leads to high buffer clients getting stuck in quality levels greater than their fair share and preventing lower quality level clients from switching up. This can be seen in Fig. 4.3 where Client 1 switches to higher quality levels aggressively around 80 seconds and gets stuck in a high quality level. This is present in every network scenario because they are not long enough for buffer levels to drop below 2 seconds allowing clients to switch to lower quality levels.

Utilization and Quality: The rate-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels. The rate-based algorithm produced the highest utilization due to clients requesting high quality levels during the initial phase. This can be seen in Fig. 4.5 where clients request very high quality levels during the initial phase. This is the result of requesting the lowest quality for the first segment and a non-full harmonic mean window that leads to over estimations of bandwidth as shown in Chapter 3. This is present across network scenarios because each network scenario is identical for the first 15 seconds. FESTIVE produced the lowest utilization and quality levels due to ramp-up behavior.

This can be seen in 4.4 where each client spends the majority of the stream ramping up to the appropriate quality level.

Switches: The resolution-based algorithm produced the lowest number of quality switches and the buffer-based algorithm produced the highest number of quality switches. The resolution-based algorithm calculates a fair quality for each client based on the clients resolution and the average bandwidth for the network scenario. This solution is calculated using static values, and remains constant throughout the stream, resulting in a low number of quality switches. The average bandwidth for each network scenario is very similar which leads to this trend being present across scenarios. As shown in the pseudo-code in Chapter 3, the buffer-based algorithm calculates a new bitrate whenever $B_{k-1} - B_{k-2} \neq 0$ where B_{k-1} and B_{k-2} are the buffer levels at $k-1$ and $k-2$ respectively. This means any small change in buffer level leads to a potentially new quality level, resulting in a high number of switches. This may not occur if bandwidth drops were not large enough to cause quality switches however, we have designed each network scenario to produce quality switches which is why this is present across all scenarios.

Buffer: FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. FESTIVE produced the highest buffer levels due to long ramp-up behavior that results in very low bandwidth utilization and leads to high buffer levels. As explained previously, the high and low buffer thresholds of the hybrid algorithm result in clients getting stuck in high quality levels resulting in low buffer levels.

Initial Phase Length: The resolution-based and hybrid algorithms produced the shortest initial phases and the the rate-based algorithm produced the longest initial phases. Both the resolution-based and hybrid algorithms request the lowest quality segments during the initial phase resulting in average initial phase lengths of 0.6 seconds. This behavior is unaffected by the network scenario as each network scenario has the same bandwidth limit for the first 0.6 seconds. FESTIVE also requests the lowest quality segments during the initial phase however, the random delay added to each segment request leads to slightly longer initial phases. We have already explained how the

rate-based algorithm has a tendency to over estimate bandwidth during the initial phase leading to high quality levels. This tendency also results in very long initial phases due to the long segment download times.

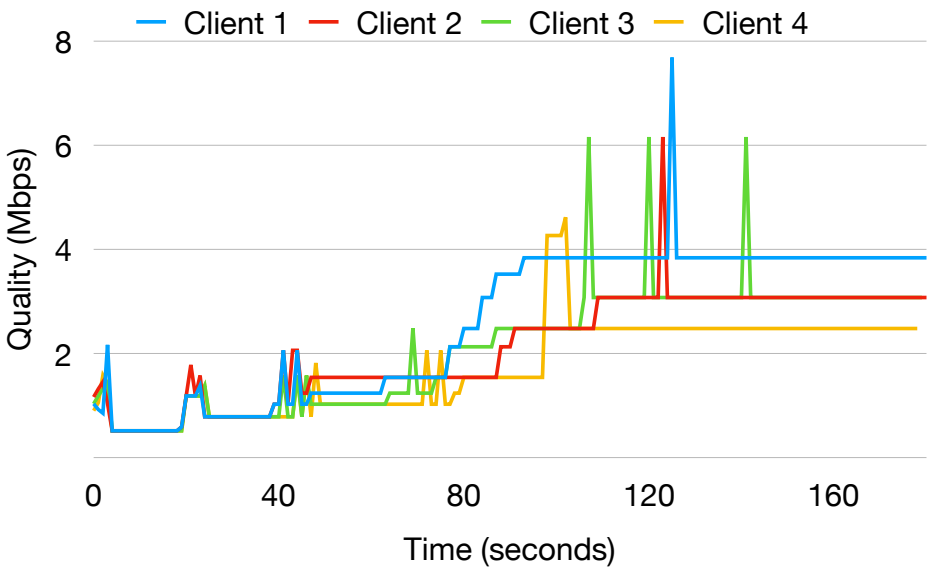

Figure 4.3: Quality levels for the hybrid algorithm during the Drop 1 network scenario under the homogeneous client scenario.

4.2.2 Mixed Client Scenarios

Under the homogeneous client scenarios we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) FESTIVE produced the best fairness and the hybrid algorithm produced the worst fairness. (2) The rate-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels. (3) The resolution-based algorithm produced the lowest number of switches and the buffer-based algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

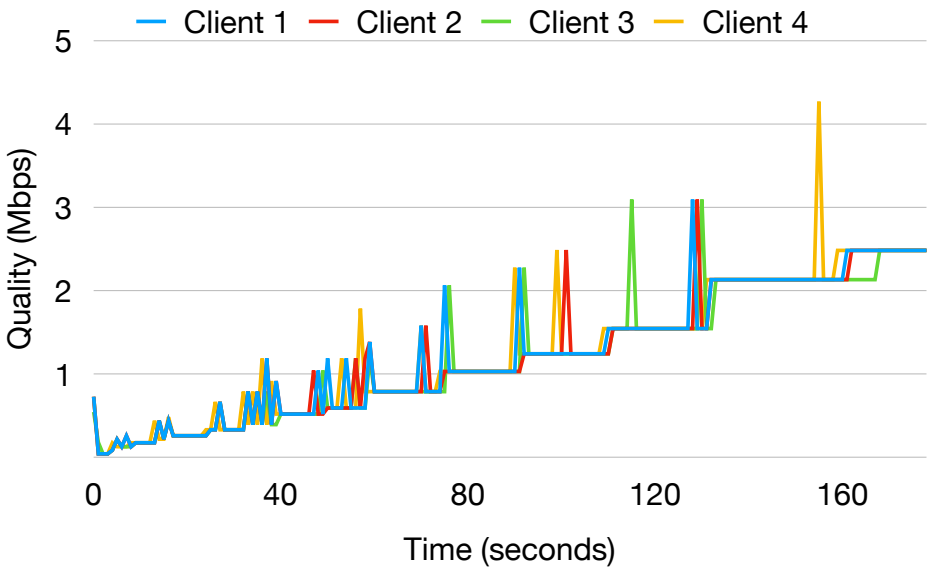

Figure 4.4: Quality levels for FESTIVE during the Drop 1 network scenario under the homogeneous client scenario.

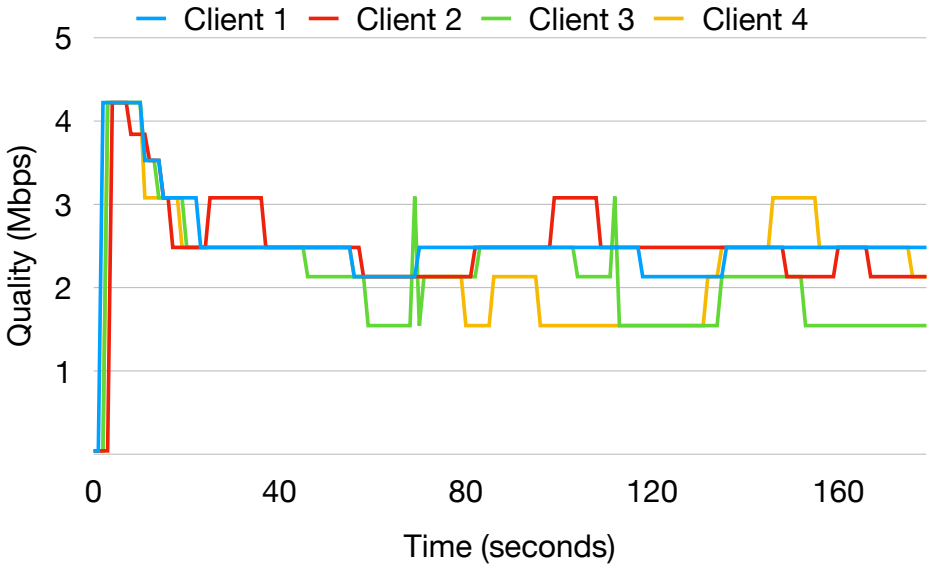

Figure 4.5: Quality levels for the rate-based algorithm during the Drop 3 network scenario under the homogeneous client scenario.

These 5 trends are largely repeated under the mixed client scenario as shown in Table 4.2. The results in their entirety can be seen in in Tables A.5 to A.8 in Appendix A. For the sake of brevity, we will only focus on the trends that are different from the homogeneous client scenarios.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.005653**	**74.8**	**1.2**	28.6	**0.6**	**1.963**
Buffer	0.009671	68.1	10.7	28.6	10.1	1.789
Rate	0.007309	73.6	5.7	28.4	24.5	1.932
FESTIVE	0.032230	25.2	11.0	**29.4**	1.2	0.661
Hybrid	0.022612	40.7	13.0	24.6	**0.6**	1.068

Table 4.2: Results from the Drop 1 network scenario under a mixed client scenario.

Fairness: The resolution-based algorithm produced the best fairness and FESTIVE produced the worst fairness. The resolution-based algorithm uses SSIM values and client resolutions to determine quality levels. This allows the resolution-based algorithm to achieve achieve QoE fairness between resolution groups. In contrast to this, FESTIVE is good at producing intra-group fairness, as shown in the homogeneous client scenarios, but produces very low inter-group fairness. These tendencies are independent of network scenario which is why this trend occurs across network scenarios.

Utilization and Quality: The resolution-based algorithm produced the highest utilization and quality levels. The resolution-based algorithm calculates a fair quality level, f_{q_i}, for each client i. Upon every request, r_i, the algorithm returns $min(f_{q_i}, r_i)$. This means the quality level of client i can never exceed f_{q_i}. Under the homogeneous client scenario, each clients f_{q_i} was 2.5 Mbps meaning each client was unable to request the 4 highest quality levels. This lead to lower utilization than the rate-based clients. Under the mixed client scenario, each clients f_{q_i} is set much higher with f_{q_i} being set to the maximum quality level under drop 1 and drop 2. This leads to higher utilization in the mixed client scenarios. The other contributing factor is requesting the lowest quality segments during the initial phase.

It is worth noting that rate-based and resolution-based clients are nearly identical with the only difference being that resolution-based clients request the lowest quality segments during the initial

phase. When introducing the midway quality strategy in Chapter 3, we showed that requesting the lowest quality segments during the initial phase leads rate-based clients to accumulate a large number of high throughput values due to the very small segment sizes. The smaller segment sizes and the higher f_{q_i} values of the mixed client scenario result in the resolution-based algorithm having high quality levels and utilization.

Switches: The hybrid algorithm produced the highest number of switches. Under the homogeneous client scenario, the network scenarios are able to support quality levels around 2.5 Mbps for each client. This would require 15 quality switches to reach from the lowest quality level when switching one level at a time like the hybrid algorithm. Under the mixed client scenarios, the network scenarios provide more than enough bandwidth to support maximum quality levels of 4.2 Mbps for the 1920x1080 clients and 396 Kbps for 480x360 clients. This would require 19 quality switches and 7 quality switches to reach from the lowest quality level for the 1920x1080 and 480x360 clients respectively. The ability of network scenarios to provide more than enough bandwidth to support maximum quality levels also leads to more steady buffer levels resulting in fewer switches for the buffer-based algorithm. This ultimately leads to the hybrid algorithm having the highest number of switches.

4.2.3 Examining the 1080p Clients Under Mixed Client Scenarios

In this section will we re-analyze the results from the mixed client scenario tests but examine each resolution (1920x1080 and 480x360) separately to get homogeneous client scenario results within the mixed client scenarios. We will present the 1920x1080 results first in this section followed by the 480x360 results in the following section. We expect that the results will be very similar to the homogeneous client scenarios because the clients are homogeneous within each group.

Under the homogeneous client scenarios, we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) FESTIVE produced the best fairness and the hybrid algorithm produced the worst fairness. (2) The rate-based algorithm produced the highest utilization and

quality levels and FESTIVE produced the lowest utilization and quality levels. (3) The resolution-based algorithm produced the lowest number of switches and the buffer-based algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

The same 5 trends as the homogeneous client scenarios are largely present across network scenarios with slight differences as shown in Table 4.3. The results in their entirety can be seen in Tables A.9 to A.15 in Appendix A. We will focus our analysis on differences between client scenarios.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00040102	**68.2**	**1.3**	28.3	**0.6**	**3.583**
Buffer	0.00020484	60.6	19.3	27.7	15.8	3.183
Rate	0.00062640	66.1	10.3	27.4	43.8	3.469
FESTIVE	**0.00007297**	18.8	15.0	**29.2**	1.2	0.986
Hybrid	0.00101045	34.4	19.0	24.5	0.7	1.808

Table 4.3: Results from the Drop 1 network scenario under a mixed client scenario showing only the 1080p clients.

Utilization and Quality Levels: The resolution-based algorithm produced the highest utilization and quality levels. This is because the mixed client scenario affects the magnitude of the supported quality levels of the 1080p clients compared to homogeneous client scenarios. Under homogeneous clients scenarios, each of the 4 clients had a resolution of 1920x1080 and quality levels for each client tended to be around 2.5 Mbps. Under the mixed client scenarios, only 2 of the 4 clients had a resolution of 1920x1080 and quality levels for these clients tended to be around 4.2 Mbps. This means that each 1080p client i under homogeneous and mixed client scenarios have a different f_{q_i} leading to different limits on quality levels. This is why the same trend from the mixed client scenarios is present here.

Switches: The buffer-based algorithm produced the highest number of switches. It is worth noting that the hybrid algorithm produced the highest number of switches under the mixed

client scenarios. This is because the hybrid algorithm produced a higher number of quality switches for the 480x360 clients than the buffer-based algorithm due to ramp-up behavior. This can be seen by comparing the quality levels of 360p clients under drop 1 of the hybrid algorithm in Fig. 4.6 and the buffer-based algorithm in Fig. 4.7. Notice that the hybrid algorithm has a longer ramp-up phase than the buffer-based algorithm. This leads to the hybrid algorithm having a higher number of switches overall however, the buffer-based algorithm produced a higher number of switches for 1080p clients.

4.2.4 Examining the 360p Clients Under Mixed Client Scenarios

The bandwidth during each scenario was far above the required amount for each 480x360 client to stream at the maximum quality level of 396 Kbps, thus the differences between algorithm performance are very small. For each algorithm, each 480x360 client spent the vast majority of the stream at the maximum quality level of 396 Kbps with the only differences between algorithms being initial phase behavior and ramp-up behavior for FESTIVE and the hybrid algorithm. This can be seen by comparing the quality levels between the buffer-based algorithm in Fig. 4.7 and the hybrid algorithm in Fig.4.6. For this reason will not present the full results of the 480x360 clients.

In summary, this section tested each algorithm under a variety of network scenarios that consist of bandwidth drops at different frequencies and magnitudes under the homogeneous and mixed client scenarios. We found that the client scenario had a greater impact on results than the network scenario. Across all scenarios, we found the following positive trends: (1) the resolution-based algorithm produced the lowest number of quality switches, (2) FESTIVE produced the highest buffer levels, and (3) the resolution-based algorithm produced the shortest initial phases. We also found the following negative trends: (1) FESTIVE produced the lowest utilization and quality levels, (2) the hybrid algorithm produced the lowest buffer levels, and (3) the rate-based algorithm produced the longest initial phases. Fairness was the metric most impacted by different client scenarios with FESTIVE producing the best fairness under homogeneous client scenarios and the resolution-based algorithm producing the best fairness under mixed client scenarios.

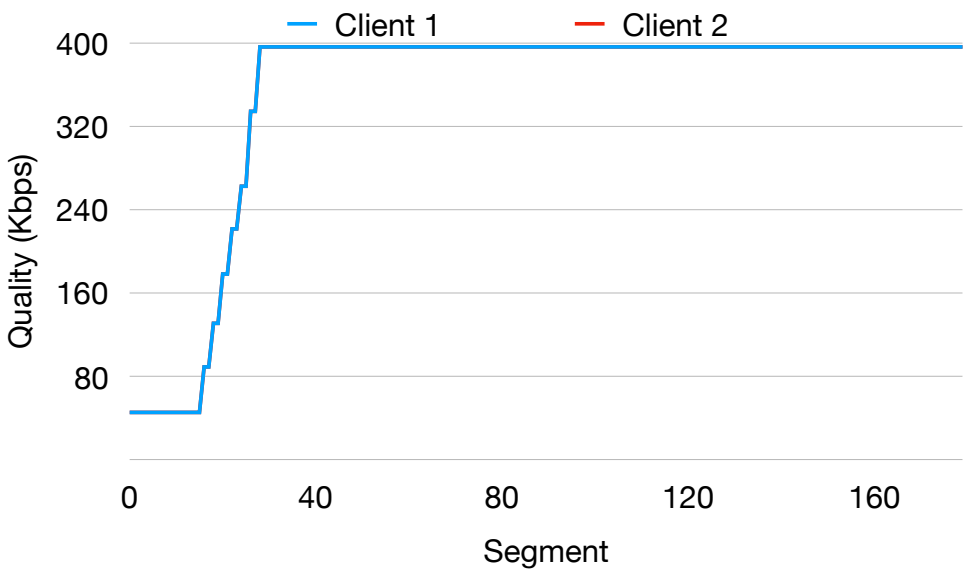

Figure 4.6: Quality levels of 360p hybrid clients during the Drop 1 network scenario under the homogeneous client scenario.

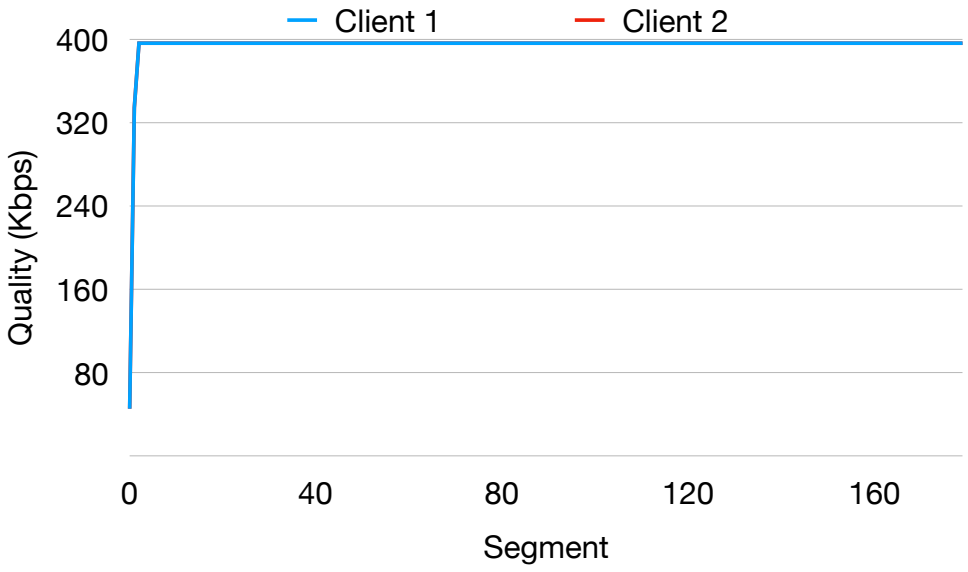

Figure 4.7: Quality levels of 360p buffer-based clients during the Drop 1 network scenario under the homogeneous client scenario.

4.3 Bandwidth Drops and Raises

This section will test each algorithm under 3 different network scenarios, referred to as big drop, big raise, and drop and raise, as shown in Fig. 4.8. A bandwidth limit of 11 Mbps should allow each client to stream at a quality of 2.5 Mbps and a bandwidth limit of 9.5 Mbps should allow each client to stream at a quality of 2.1 Mbps under homogeneous client scenarios. A bandwidth limit of 11 Mbps and 9.5 Mbps should allow both groups to stream at their maximum quality levels of 4.2 Mbps and 396 Kbps respectively under mixed client scenarios. A bandwidth limit of 8 Mbps should allow each client to stream at a quality of 1.5 Mbps under homogeneous client scenarios and 1920x1080 clients to stream at a quality of 3.5 Mbps and 480x360 clients to stream at a quality level of 396 Kbps under mixed client scenarios. We will first examine the results of all homogeneous client scenario tests followed by the results of the mixed client scenario tests and finish with the 1080p and 360p clients under mixed client scenarios.

4.3.1 Homogeneous Client Scenarios

Under the previous homogeneous client scenarios we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) FESTIVE produced the best fairness and the hybrid algorithm produced the worst fairness. (2) The rate-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels. (3) The resolution-based algorithm produced the lowest number of switches and the buffer-based algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

These 5 trends are present across all network scenarios with the exception of fairness, where the resolution-based, hybrid, and buffer-based algorithms produced the worst fairness under the

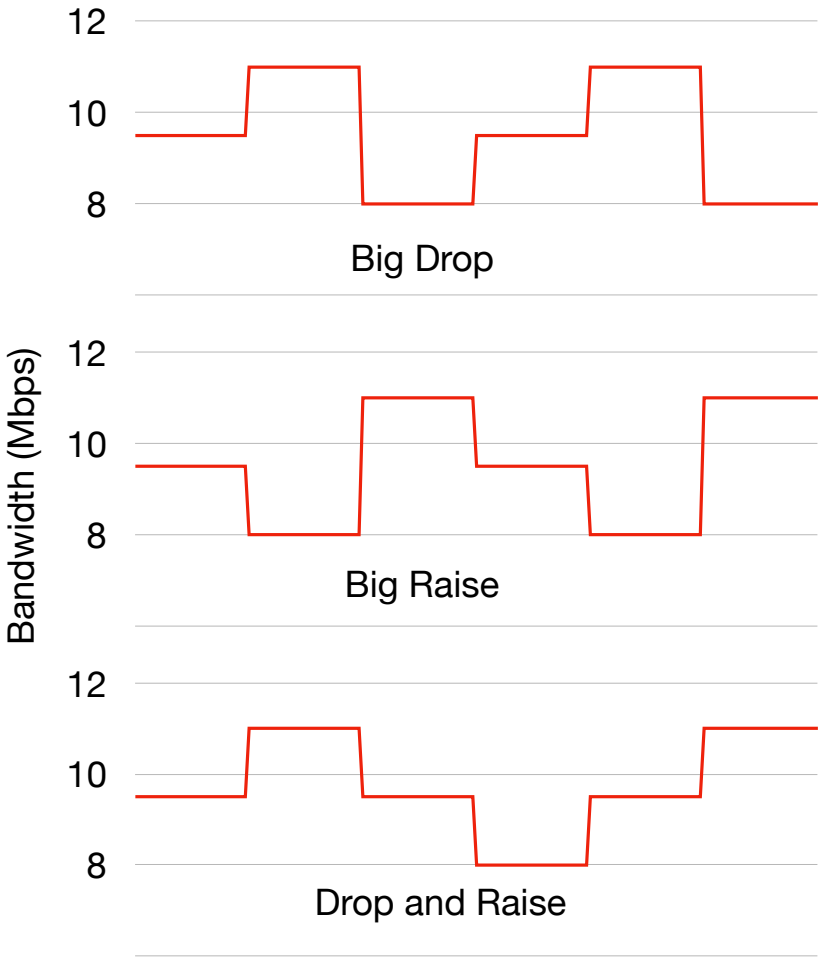

Figure 4.8: Big Drop, Big Raise, and Drop and Raise network scenarios.

big drop, big raise, and drop and raise network scenarios respectively. These trends can be seen in the results of the Big Drop network scenario shown in Table 4.4. The results in their entirety can be seen in Tables A.17 to A.19 in Appendix A.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.002014	78.8	**3.3**	27.7	**0.7**	1.904
Buffer	0.000837	86.8	22.9	26.2	18.4	2.097
Rate	0.000889	**93.6**	10.1	25.7	40.0	**2.263**
FESTIVE	**0.000519**	37.4	15.5	**28.9**	1.4	0.903
Hybrid	0.001496	56.1	16.2	24.2	**0.7**	1.356

Table 4.4: Results from the Big Drop network scenario under a homogeneous client scenario.

Fairness: The resolution-based, hybrid, and buffer-based algorithms produced the worst fairness under the big drop, big raise, and drop and raise network scenarios respectively. It is worth noting that the average bandwidth is the same between the big drop and big raise network scenarios. The resolution-based algorithm limits the quality level of each client i to a fair quality level, f_{q_i}, calculated at the start of the stream. f_{q_i} is calculated based on client resolutions, SSIM values, and the average bandwidth of the network scenario. Under the big drop and big raise network scenarios, there is not sufficient bandwidth for each client to stream at 2.5 Mbps, thus two clients have their $f_{q_i} = 2.5$ Mbps and the other two have their $f_{q_i} = 2.1$ Mbps. The problem is made worse due to large drops in bandwidth increasing the number of potential quality levels where large increases in bandwidth do not due to the quality limit of f_{q_i}.

We have previously explained how the hybrid algorithm leads to poor fairness due to high buffer clients getting stuck in high quality levels. This problem is made worse by large bandwidth raises, resulting in the worst fairness under the big raise network scenario. The buffer based algorithm produced the worst fairness under the drop and raise network scenario due to the high bandwidth limit within the first 90 seconds compared to the other network scenarios. This leads to some clients gathering a large number of high throughput values and using more than their fair share of bandwidth. This can be seen in Fig. 4.9, where clients 2 and 4 end up using more than their fair share around the 75 second mark.

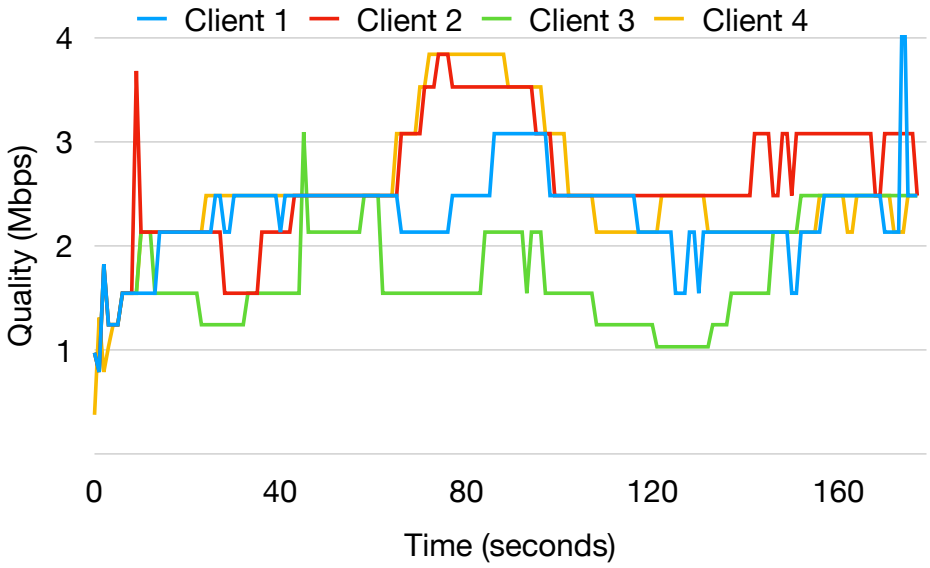

Figure 4.9: Quality levels for the buffer-based algorithm under the Drop and Raise network scenario.

4.3.2 Mixed Client Scenarios

Under the previous mixed client scenarios we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) The resolution-based algorithm produced the best fairness and FESTIVE produced the worst fairness, (2) the resolution-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels, (3) The resolution-based algorithm produced the lowest number of switches and the hybrid algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

These 5 trends are present across all network scenarios with no exceptions. The trends can be seen in the results for the Big Drop network scenario shown in Table 4.5. The results in their entirety can be seen in Tables A.20 to A.22 in Appendix A. In the previous tests, we have explained

how algorithm performance is largely determined by mechanisms that are unaffected by the different network scenarios tested. We have also pointed out metrics that are affected by the client scenario, such as fairness, and explained how they are affected. The Big Drop, Big Raise, and Drop and Raise network scenarios are different from Drops 1 to 4 however, the differences are not significant with the average bandwidths, bandwidth levels, and frequencies of bandwidth changes being very similar. Thus, it is not surprising the results of these mixed client scenarios are very similar to the previous mixed client scenarios.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.006973**	**79.6**	**2.5**	28.5	0.8	**1.923**
Buffer	0.011472	69.5	12.8	28.4	11.3	1.680
Rate	0.009952	72.1	7.3	28.2	23.3	1.743
FESTIVE	0.032365	27.0	11.0	**29.3**	2.0	0.653
Hybrid	0.023902	40.9	13.0	24.4	**0.7**	0.988

Table 4.5: Results from the Big Drop network scenario under a mixed client scenario.

4.3.3 Examining the 1080p Clients Under Mixed Client Scenarios

When previously examining the 1080p clients under mixed client scenarios, we identified and explained 5 trends that were present across all network scenarios for 1080p clients and explained why they are not impacted by different network scenarios. These trends are as follows: (1) FESTIVE produced the best fairness and the hybrid algorithm produced the worst fairness. (2) The resolution-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels. (3) The resolution-based algorithm produced the lowest number of switches and the buffer-based algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

These 5 trends are present across all network scenarios with no exceptions. The trends can be seen in the results for the Big Raise network scenario shown in Table 4.6. The results in their

entirety can be seen in Tables A.23 to A.27 in Appendix A. Like the previous section, it is not surprising that these results are similar to the previous results due to the similarly between network scenarios and small impact that these different network scenarios have on the results. We also wont being showing the 480x360 results as they are almost identical with small differences like the previous results.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00030601	**72.1**	**6.0**	27.4	0.8	**3.484**
Buffer	0.00003619	58.7	26.2	26.7	16.6	2.837
Rate	0.00021158	62.1	13.5	27.6	41.9	3.000
FESTIVE	**0.00006911**	21.1	15.5	**29.3**	1.9	1.018
Hybrid	0.00314608	32.1	18.3	24.4	**0.7**	1.552

Table 4.6: Results from the Big Raise network scenario under a mixed client scenario showing only the 1080p clients.

In summary, this section tested each algorithm under a variety of network scenarios that consist of a mixture of bandwidth drops and raises under the homogeneous and mixed client scenarios. We found that the client scenario had a greater impact on results than the network scenario. Across all scenarios, we found the following positive trends: (1) the resolution-based algorithm produced the lowest number of quality switches, (2) FESTIVE produced the highest buffer levels, and (3) the hybrid algorithm produced the shortest initial phases. We also found the following negative trends: (1) FESTIVE produced the lowest utilization and quality levels, (2) the hybrid algorithm produced the lowest buffer levels, and (3) the rate-based algorithm produced the longest initial phases. Fairness was the metric most impacted by different client scenarios with FESTIVE producing the best fairness under homogeneous client scenarios and the resolution-based algorithm producing the best fairness under mixed client scenarios.

4.4 Real Bandwidth Trace

This section will use a real world bandwidth trace to determine how well each algorithm is able to perform under realistic network conditions. The data set was provided by Raca *et al.* and contains

throughput values obtained over a 4G cellular network from a single client at a granularity of one sample per second [45]. We chose to use this trace because it is recent (published within a year of writing this thesis) and because of the growing popularity of video streaming over cellular networks. We could have also used a number of different studies on the workload of video streaming severs [53, 3, 51] and will be sure to in any future work. For this test we will be using 20 clients, thus we have simply multiplied the throughput values by 20 to achieve the bandwidth limit shown in Fig. 4.10. We will only be using the mixed client scenario during this test, meaning 10 of the clients will have a resolution of 1920x1080 and the other 10 will have a resolution of 480x360. We will first present the results of the mixed client scenario followed by the 1080p and 360p clients under mixed client scenarios.

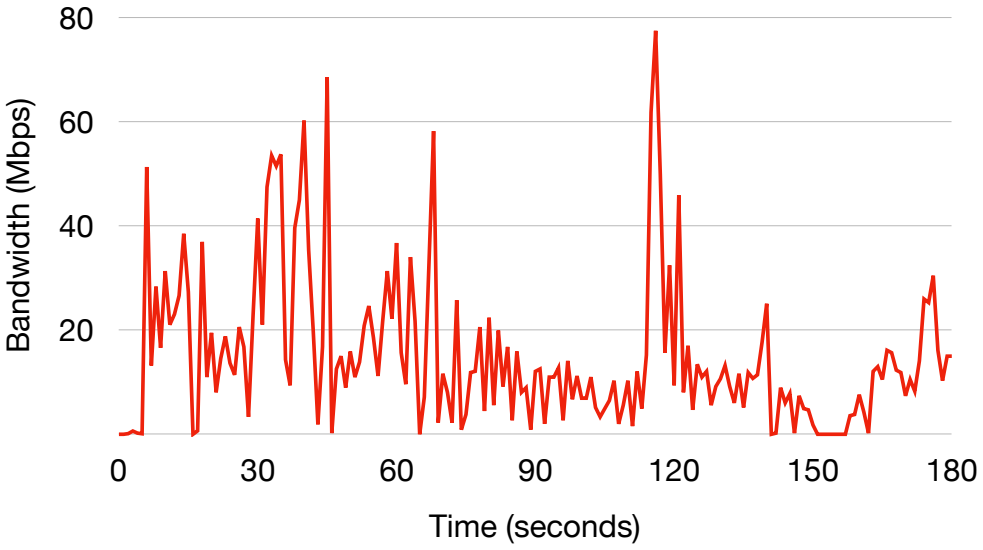

Figure 4.10: Bandwidth under the real bandwidth trace network scenario.

4.4.1 Mixed Client Scenario

Under the previous mixed client scenarios we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) The resolution-based algorithm produced the best fairness

and FESTIVE produced the worst fairness, (2) the resolution-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels, (3) The resolution-based algorithm produced the lowest number of switches and the hybrid algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

These 5 trends are largely present in the results for the real bandwidth trace, as shown in Table 4.7. We will focus our analysis on the differences between previous mixed client scenarios and how the network scenario lead to those differences.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.014651**	74.1	**4.5**	25.8	1.3	0.562
Buffer	0.038831	77.1	27.6	24.9	19.2	0.585
Rate	0.032321	**82.8**	11.6	22.6	26.3	**0.628**
FESTIVE	0.039453	46.6	14.4	**26.9**	**1.2**	0.353
Hybrid	0.039374	68.1	12.1	22.7	1.3	0.516

Table 4.7: Results from the Real Bandwidth Trace network scenario under a mixed client scenario.

Utilization and Quality Levels: The rate-based algorithm produced the highest utilization and quality levels. Under all the other mixed client scenario tests, the resolution-based algorithm produced the highest utilization and quality levels. However, under the real bandwidth trace the rate-based algorithm produced the highest utilization and quality levels. This is due to the high throughput values obtained by the 1920x1080 clients during the initial phase as a result of a non-full harmonic window and the large number of clients. During the other 7 network scenarios, the rate-based clients also experienced high throughput values during the initial phase however, those network scenarios had a smaller number of clients and were able to support quality levels much closer to the maximum quality level. This means the high throughput values during the initial phase in other network scenarios did not have a large effect on the overall utilization and quality levels. The supported quality levels of the 1920x1080 clients are much lower during this network scenario,

as shown the by low average quality levels observed during this test, thus clients streaming at 4.2 Mbps in the initial phase has a greater impact on the overall quality levels and utilization.

Switches: The buffer-based algorithm produced the highest number of quality switches. Under previous mixed client scenarios, the hybrid algorithm produced the highest number of switches due to longer ramp-up behavior than the buffer-based algorithm. This ramp-up behavior is still present however, the network scenario is much less stable than previous network scenarios resulting in much less stable buffer levels. The buffer-based algorithm is very sensitive to small changes in buffer levels resulting in the highest number of switches during the real bandwidth trace.

Buffer: The rate-based algorithm produced the lowest buffer levels. Under other network scenarios, the hybrid algorithm produced the lowest buffer levels due to clients getting stuck in high quality levels. This still occurs here (as shown by the second lowest buffer levels produced by the hybrid algorithm) however, the large number of clients and very dynamic network scenario prevent clients from switching up quality levels as aggressively as in previous scenarios. Under the real bandwidth trace, the rate-based algorithm produced the lowest buffer levels due to producing the highest utilization and quality levels. This was also made worse by using the harmonic mean of the previous 20 throughput values as a bandwidth estimation which does not react to changes in network conditions quickly.

Initial Phase Length: FESTIVE produced the shortest initial phases. FESTIVE produced the shortest initial phase due to requesting the lowest quality segments during the initial phase and the random delay added to each segment request. Under other network scenarios, the random delay of FESTIVE increased the initial phase length slightly resulting in the resolution-based or hybrid algorithm producing the shortest initial phases. Under the real bandwidth trace, the random delay decreases the chances of clients having overlapping segment downloads that has a greater impact on initial phase lengths with a large number of clients.

4.4.2 Examining the 1080p Clients Under Mixed Client Scenarios

When previously examining the 1080p clients under mixed client scenarios, we identified and explained 5 trends that were present across all network scenarios and explained why they are not impacted by different network scenarios. These trends are as follows: (1) FESTIVE produced the best fairness and the hybrid algorithm produced the worst fairness. (2) The resolution-based algorithm produced the highest utilization and quality levels and FESTIVE produced the lowest utilization and quality levels. (3) The resolution-based algorithm produced the lowest number of switches and the buffer-based algorithm produced the highest number of switches. (4) FESTIVE produced the highest buffer levels and the hybrid algorithm produced the lowest buffer levels. (5) The resolution-based and hybrid algorithms produced the shortest initial phases and the rate-based algorithm produced the longest initial phases.

These 5 trends are present with the exception of the shortest initial phase length and lowest buffer levels, as shown in Table 4.8. Under previous mixed client scenarios, the clients almost exclusively affected by the bandwidth changes were the 1920x1080 clients with the 480x360 clients mostly streaming at the maximum quality level. Under this network scenario, the 480x360 clients are affected by the network changes due to a larger number of clients and more dynamic network conditions. This is why these results are almost identical to previous 1080p clients under the mixed client scenario but the overall results have differences. We will focus our analysis on the initial phase length and buffer levels and how they are affected by the network scenario.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.001378	**59.1**	**7.9**	23.7	1.2	**0.897**
Buffer	0.003418	51.2	53.2	22.1	25.3	0.777
Rate	0.002478	57.3	18.5	17.3	37.5	0.869
FESTIVE	**0.001368**	26.7	18.0	**26.4**	**0.9**	0.404
Hybrid	0.005016	46.9	17.2	21.6	1.6	0.711

Table 4.8: Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 1080p clients.

Buffer: The rate-based algorithm produced the lowest buffer levels. In the previous sec-

tion, we explained how the real bandwidth trace prevented hybrid clients from aggressively switching up quality levels and getting stuck compared to other network scenarios. This lead the hybrid algorithm to only produce the second lowest buffer levels when it usually produces the lowest buffer levels. We also explained how the rate-based algorithm produced the lowest buffer levels due to also producing highest utilization and quality levels and using the harmonic mean of the previous 20 throughput values as bandwidth estimation. This same explanation applies to 1080p clients and is why the rate-based algorithm produced the lowest buffer levels.

Initial Phase Length: FESTIVE produced the shortest initial phases. FESTIVE produced the shortest initial phases due to random delayed added to each segment request. This reduced the chances that clients will perpetually download segments at the same time resulting in shorter segment download times. The impact of random delay is made greater by the larger number of clients and the larger segment sizes of 1920x1080 clients compared to 480x306 clients.

4.4.3 Examining the 360p Clients Under Mixed Client Scenarios

Previously, we only examined the 1920x1080 clients under mixed client scenarios due to the almost identical performance of the 480x360 clients across algorithms. This is because the previous network scenarios and low number of clients allowed the 480x360 clients to stream at the maximum quality level for the majority of the stream with the only difference between algorithms being the initial phase and ramp-up behavior. Due to the much more dynamic nature of the real bandwidth trace and the larger number of client, the 480x360 clients produced much more varied results and are worth examining separately. Since we have no previous results to compare to, we will be presenting the results in their entirety and explain the trends present in Table 4.9.

Fairness: The buffer-based algorithm produced the best fairness and FESTIVE produced the worst fairness. The buffer-based algorithm is very sensitive to bandwidth changes which is shown in the high number of quality switches of the 1920x1080 clients. This extreme sensitivity results in the 1920x1080 reacting to all of the network changes present in the real bandwidth trace. By being very reactive to network changes, the 1920x1080 clients reduce the negative impact they

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000716	15.0	**1.2**	**27.9**	1.4	0.228
Buffer	**0.000129**	**25.9**	2.1	27.7	13.1	**0.393**
Rate	0.000995	25.5	4.7	**27.9**	15.2	0.387
FESTIVE	0.002000	19.9	10.9	27.3	1.5	0.302
Hybrid	0.000923	21.2	7.0	23.7	**1.1**	0.322

Table 4.9: Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 360p clients.

have on the 480x360 allowing them to stream at a more constant quality level. The long ramp-up behavior of FESTIVE lead to good fairness in the 1920x1080 clients however, the much shorter ramp-up behavior of the 480x360 clients decreases fairness. FESTIVE uses 85% of the bandwidth estimation to determine if a client should switch to lower quality level. This means FESTIVE clients are more sensitive to drastic drops in bandwidth that are present in the real bandwidth trace. This ultimately leads to greater variation in quality levels and worse fairness.

Utilization and Quality Levels: The buffer-based algorithm produced the highest utilization and quality levels and the resolution-based algorithm produced the lowest utilization and quality levels. We have previously explained that the fair quality level calculated by the resolution-based algorithm limits 480x360 clients to below the maximum of 396 Kbps resulting in low utilization. We have also previously explained how the 1920x1080 buffer-based clients are very reactive to network changes that reduces the negative impact on the 480x360 clients. This results in the buffer-based 480x360 clients having the highest utilization and quality levels.

Switches: The resolution-based algorithm produced the lowest number of quality switches and FESTIVE produced the highest number of quality switches. We have previously explained how the resolution-based algorithm leads to the lowest number of quality switches due to using a constant solution. FESTIVE produced the highest number of quality switches due to ramp-up behavior and using 85% of the bandwidth estimation to determine if a client should switch to lower quality level. This means FESTIVE clients are more sensitive to drastic drops in bandwidth that are present in the real bandwidth trace.

Buffer: The hybrid algorithm produced the lowest buffer levels while all the other algorithms produced almost identical buffer levels. We have previously explained how hybrid clients can only switch to lower quality levels when they have buffer levels below 2 seconds. This prevents them from adapting to drops in bandwidth and leads to low buffer levels. Every other algorithm allows clients to switch to lower quality levels at any buffer level allowing them to adapt to bandwidth drops and leading to higher buffer levels. The reason all the buffer levels are so similar is that the bandwidth limit is more than sufficient for the 480x360 clients to stream at the maximum quality for the majority of the stream with a severe drop of bandwidth around 150 seconds in. For the majority of the stream, all 480x360 clients had very high buffer levels with large drops around 150 seconds resulting in very similar overall buffer levels.

Initial Phase Length: The hybrid algorithm produced the shortest initial phases and the rate-based algorithm produced the longest initial phases. We have previously explained how the rate-based algorithm produced the longest initial phases due to high quality levels during the initial phase. For the 1920x1080 clients, FESTIVE produced the shortest initial phases as a result of the random delay added to each segment request. This does not have as great of an impact with the 430x360 clients due to the much smaller segment sizes and leads to FESTIVE having a slightly longer initial phase. The resolution-based algorithm also requests the lowest quality segments during the initial phase however, the slight overhead introduced by the server and the large number of clients increases the initial phase slightly. This results in the hybrid algorithm having the shortest initial phases for the 480x360 clients.

In summary, this section tested each algorithm with a real bandwidth trace under the mixed client scenario. Overall, we found that the resolution-based algorithm performed well producing the best fairness, the lowest number of switches, short initial phases, and near the highest buffer levels. Upon examining each resolution group separately, we found that the resolution-based algorithm left room for improvement in terms of fairness and utilization with resolution groups.

4.5 Summary

In this Chapter we performed a more detailed cross comparison of DASH adaptation algorithms using 8 different network scenarios and 2 client scenarios. The choice of algorithms was based on the results of the analysis done in the previous Chapter and include a rate-based, buffer-based, hybrid, server-side fairness, and client-side fairness algorithm. The network scenarios used are more varied and realistic than the scenarios used in Chapter 3 and consist of various different frequencies and magnitudes of bandwidth drops, a mixture of bandwidth drops and raises, and a real bandwidth trace. The client scenarios consist of a homogeneous client scenario where each client has the same screen resolution, and a mixed client scenario where half of the clients have a 1920x1080 resolution and the other half have a 480x360 resolution.

Overall, we found that test results were largely determined by mechanisms that are independent of network scenarios resulting in client scenarios having a greater impact on results than network scenarios. The greatest impact of the client scenarios was fairness, utilization, and quality levels with the resolution-based algorithm performing the best under mixed client scenarios due to the use of SSIM and client resolutions. To summarize the results we will present the pros and cons of each algorithm that can be seen in Table 4.10.

	Pros	Cons
Resolution	• Best fairness, utilization, and quality levels under mixed client scenarios • Lowest number of switches overall • Short initial phase length	• Low visual quality during initial phase • Low intra-group fairness • Low utilization for 360p clients under mxied client scenarios
Buffer	• Adaptive to network changes	• Highest number of switches overall
Rate	• High utilization and quality levels	• Longest initial phases • Lowest buffer levels under the real bandwidth trace
FESTIVE	• Best fairness under homogeneous client scenarios • Highest buffer levels overall	• Lowest utilization and quality levels overall • Worst fairness under mixed client scenarios
Hybrid	• Short initial phase length	• Lowest buffer levels overall • Lowest fairness under homogeneous client scenarios • Low visual quality during initial phase

Table 4.10: The pros and cons of each algorithm over all the tests done in this chapter.

Chapter 5

FairQ Adaptation Scheme

In Chapter 3, we performed an analysis of QoE and fairness in DASH that utilized a rate-based, a buffer-based, and a hybrid algorithm. We also utilized a client-side algorithm to achieve resource fairness, FESTIVE, and a server-side QoE fairness algorithm, resolution-based. In Chapter 4 we performed comparison of the adaptation algorithms presented in Chapter 3 using a wider variety of more realistic network and client scenarios than Chapter 3. The key observations from Chapters 3 and 4 inspired the design of FairQ, a new adaptation scheme for QoE enhancement and fairness assurance in DASH. In this Chapter, we will present the design of FairQ. Section 5.1 will list the design objectives of FairQ and the preferred method to achieve those design objective based on Chapter 3 and 4. Section 5.2 will present an overview of the design of FairQ. Section 5.3 will revisit the design objectives of FairQ and show how FairQ achieves them. Section 5.4 will present a space and time complexity analysis of FairQ. Section 5.5 will present the implementation and discuss implementation issues of FairQ. Section 5.6 will conclude this Chapter with a brief summary.

5.1 Design Objectives

In this section we will list the design objectives of FairQ and explain the preferred method to achieve those objectives based on the results from Chapters 3 and 4. We will also note any limitations of the preferred methods suggested. The design objectives of FairQ are as follows:

Objective 1: Produce QoE fairness and not simply resource fairness between clients. In Chapter 4, FESTIVE produced the best fairness under homogeneous client scenarios at the cost of low utilization and the resolution-based algorithm produced the best fairness under mixed client scenarios by using SSIM and client resolutions. Due to the low utilization of FESTIVE and the fact that mixed client scenarios (client have different resolutions) are more realistic than the ho-

mogeneous client scenarios (clients have the same resolutions), we consider the resolution-based algorithm to be the best method to achieve this design objective. The limitations of this approach are that (1) SSIM values need to be pre-determined by the server prior to hosting a video, (2) a server-side adaptation algorithm is needed, (3) clients need to notify the server of their screen resolution, (4) low visual quality during the initial phase, (5) use of a constant solution can prevent clients from reaching optimal quality levels, (6) low intra-group fairness, and (7) low utilization for 360p clients under mixed client scenarios. Like the resolution-based algorithm, FairQ will use SSIM and client resolutions to achieve QoE fairness however, FairQ will not use a constant solution to determine quality levels and should be able to achieve better fairness.

Objective 2: Produce high bandwidth utilization and quality levels without sacrificing buffer levels. In Chapter 4, the rate-based algorithm produced the highest utilization and quality levels under homogeneous client scenarios while the resolution-based produced the highest utilization and quality levels under mixed client scenarios. Both of these algorithms have clients that use the harmonic mean of the previous 20 throughput values as bandwidth estimation which was shown to produce high utilization in Chapter 3. For these reasons we consider using the harmonic mean of the previous 20 throughput values as bandwidth estimation as the preferred method for achieving this objective. The limitations of this objective are that it can lead to long and unstable initial phases. FairQ will take a similar approach to the resolution-based and rate-based algorithms by having clients that use the harmonic mean of the previous 20 throughput values as bandwidth estimation however, FairQ will also use the midway quality strategy to prevent long and unstable initial phases.

Objective 3: Produce a low number of quality switches while still achieving adaptability and maintaining visual quality. In Chapter 4, the resolution-based algorithm produced the lowest number of quality switches across nearly all network and client scenarios. This is due to the use of a constant solution on the server and the use of the harmonic mean of the previous 20 throughput values as a bandwidth estimation on the client side. For this reason, we consider

using the harmonic mean of the previous 20 throughput values as bandwidth estimation and using a constant solution to be the preferred method for achieving this objective. The limitation of this method is that using a constant solutions can lead to reduced adaptability resulting in lower utilization and quality levels depending on the network and client scenario. FairQ will utilize the harmonic mean of the previous 20 throughput values on the client side but will not use a constant solution to achieve greater adaptability than the resolution-based algorithm.

Objective 4: Produce healthy buffer levels without sacrificing quality levels and utilization. There is no benefit to producing the highest buffer levels possible, especially when quality levels and utilization suffer as a result, and lower QoE only happens when buffer levels reach 0 and video playback stalls. In Chapter 4, FESTIVE produced the highest buffer levels across almost all client and network scenarios at the cost of low utilization and quality levels meaning it is not the preferred method. There was no algorithm that produced the 2nd highest buffer levels across most scenarios and all the other algorithms mostly produced adequate buffer levels. FairQ contains a server-side side algorithm, thus we can look at what the other server-side algorithms did to achieve adequate buffer levels. The resolution-based algorithm prevented clients from receiving quality levels higher than they requested while the buffer-based algorithm made it more difficult for clients with buffer levels below a threshold to increase their quality levels.

In Chapter 3, we showed that on average the bandwidth needed to support a quality level is 6.5% higher than the bitrate listed in the MPD that we refer to as the 'overhead of DASH'. We also showed that failing to consider the overhead of DASH can lead to low buffer levels that can eventually reach 0 and cause stalls in video playback. For these reasons we consider (1) the overhead of DASH, (2) guaranteeing that clients never receive quality levels greater than they requested, and (3) using a low buffer threshold to make quality decisions the preferred method to achieve this objective. The limitations of this method are (1) possible lower utilization due to considering the overhead of DASH, and (2) algorithm performance will be impacted by the value of the low buffer threshold. FairQ will incorporate these methods to achieve healthy buffer levels

and multiple low buffer thresholds will be tested to determine the impact on performance.

Objective 5: Produce a short and stable initial phase without sacrificing visual quality. In Chapter 4, the resolution-based and hybrid algorithms produced the shortest and most stable initial phases at the cost of low visual quality by requesting the lowest quality segments during the initial phase meaning that is not the preferred method. In Chapter 3, we presented the midway quality strategy that achieved a compromise between visual quality and length during the initial phase by producing stable initial phases that have a higher visual quality than the resolution-based or hybrid algorithms. For this reason, we consider the midway quality strategy to be the preferred method for achieving this objective. The limitation of this approach is that is requires a server-side adaptation algorithm however, FairQ already contains a server-side adaptation algorithm so this is not an issue for FairQ.

5.2 Design Overview of FairQ

In this section we will give an overview of the design of the FairQ algorithm. FairQ is a client-server algorithm that determines clients quality levels based on clients resolutions, buffer levels, request qualities, server bandwidth, and SSIM values. FairQ operates in two distinct phases that are (1) the initial phase and (2) the adaptation phase with an optional transition phase if quality smoothing is turned on as shown in Fig. 5.1 (quality smoothing will be explained in detail later in this section) . The initial phase is the time between the first segment request and time when video playback begins. The goals during the initial phase are to make the initial phase short and stable without sacrificing visual quality. This is achieved by using the midway quality strategy during the initial phase.

Along with the first request, each client sends their screen resolution to the server as shown in Fig. 5.2. After each clients first request, FairQ groups clients based on their resolutions and sends them the midway quality for the entirety of the initial phase as shown in Fig. 5.3. Notice that clients no longer send their screen resolution and instead send their buffer levels along with

Figure 5.1: Showing the initial phase, optional transition phase, and adaptation phase.

each request that will be used to determine when the client leaves the initial phase and to determine quality levels during the adaptation phase. The midway quality for each resolution group k, mid_k, is defined as $\frac{f_k}{2}$ where f_k is the fair quality level for group k. The fair quality level for group k is defined as:

$$f_k = \frac{R_k}{\sum_k n_k * R_k} \times B \tag{5.1}$$

Where n_k is the number of client in a particular resolution group, R_k is the bitrate of the lowest quality video available at resolution k, and B is 93.5% of the servers bandwidth. For client i in group k, we can choose the highest quality level that does not surpass f_k to get the fair quality level of client i, f_{q_i}. For each client i, we can pick the highest quality level that does not surpass $\frac{f_{q_i}}{2}$ to obtain the midway quality level for that client.

A client leaves the initial phase when it has received enough segments to reach a buffer level of 100% which the server can determine using the buffer levels sent along with each request shown in Fig. 5.3. If quality smoothing is turned on, the client enters the optional transition phase that transitions the clients from the midway quality to the target quality determined by the FairQ

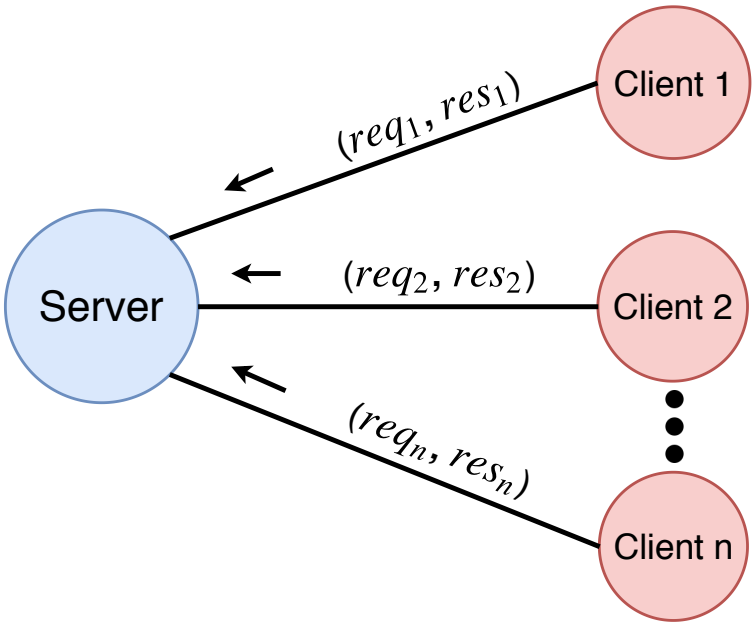

Figure 5.2: Showing the first request of n clients where each client sends their screen resolution along with the segment quality.

algorithm one quality level at a time. If quality smoothing is turned off, the client immediately jumps from the midway quality level to the target quality determined by the FairQ algorithm. Once a client leaves the transition phase, if quality smoothing is on, or the initial phase, if quality smoothing is off, they enter the adaptation phase.

During the adaptation phase, the FairQ algorithm is run upon every segment request. The algorithm starts by setting the quality level of each client i to $q_i = min(r_i, f_{q_i})$ where r_i is the most recent request quality for client i. The algorithm then calculates an average SSIM value for each resolution group based on screen resolutions and q_i for each client i. The use of SSIM and client resolutions will allow FairQ to achieve QoE fairness like the resolution-based algorithm however, FairQ does not use a constant solution that will allow FairQ to achieve better fairness than the resolution-based algorithm.

FairQ then loops through resolution groups from lowest to highest average SSIM. At each resolution group k, FairQ loops through clients and raises q_i of client i by one level if $r_i > q_i$ and $buff_i > buff_{low}$ where $buff_i$ is client is quality level and $buff_{low}$ is the low buffer threshold as

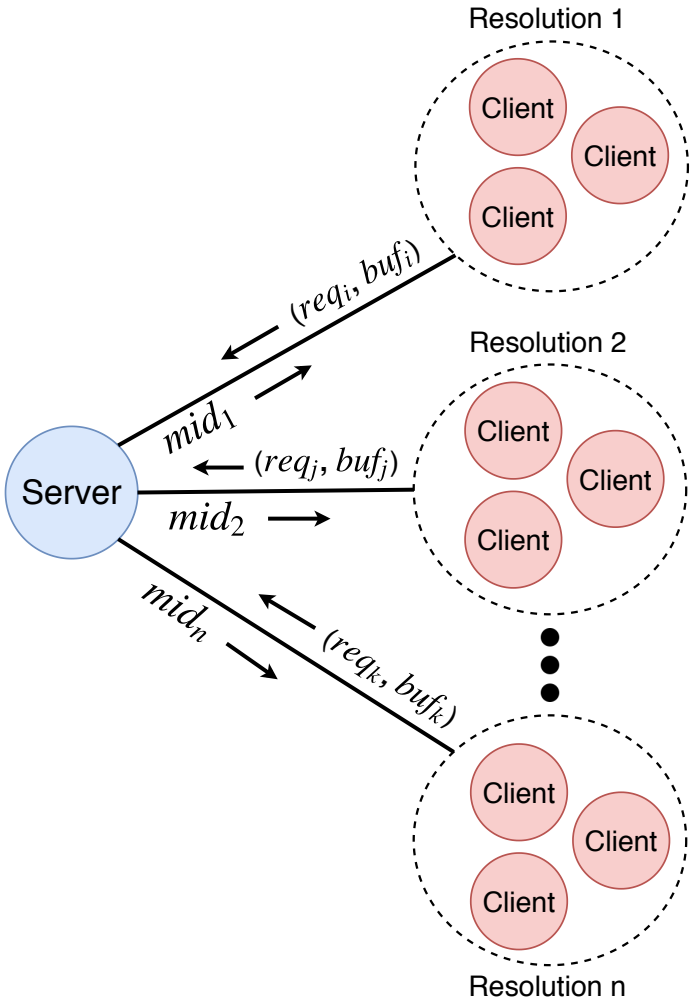

Figure 5.3: Clients during the initial phase where FairQ groups clients based on their screen resolutions and sends them the midway quality level.

shown in Fig. 5.4. By never raising a clients quality level higher than the quality they requested and only raising the quality level of high buffer clients, FairQ is able to achieve healthy buffer levels. If quality smoothing is turned on and the final quality level calculated by FairQ is greater than one quality level away, then the client will transition to the new quality level one quality level at a time. If the final quality level changes during the transition then the client will simply begin transitioning to the new final quality level. If quality smoothing is turned off then the client will simply jump to the new final quality level from its current quality level.

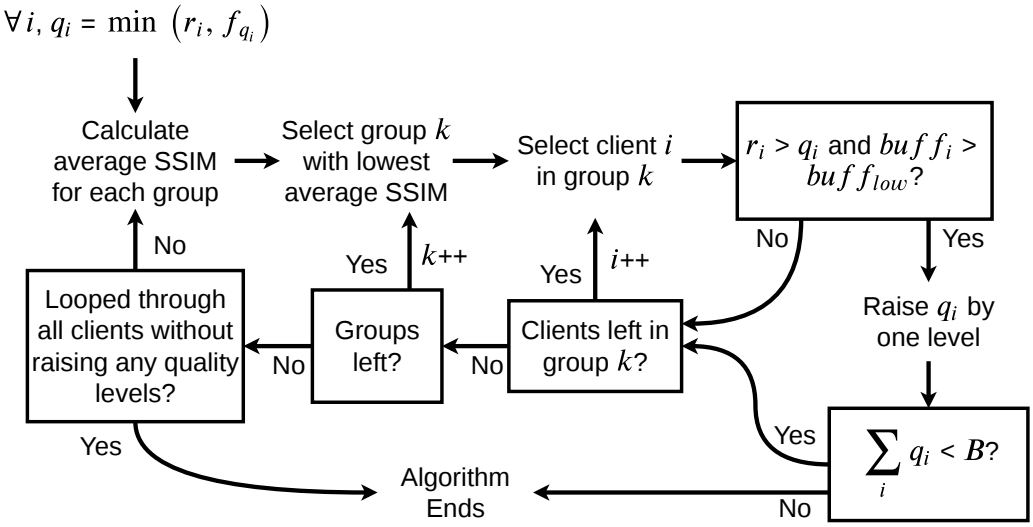

Figure 5.4: Overview of the FairQ algorithm showing how FairQ loops through resolution groups based on average SSIM values and raises quality levels of high buffer clients.

Once FairQ has looped through all clients and all groups, the average SSIM of each group is re-calculated and the algorithm repeats. The algorithm terminates when (1) there is not enough server bandwidth to raise client quality levels further, (2) all clients are at their maximum quality levels, or (3) there are no clients left that meet the criteria for having their quality levels raised. It is worth noting that FairQ does not place an upper limit on client quality levels like the resolution-based algorithm and client quality levels are generally limited by (1) server bandwidth, (2) the maximum quality level of the video being streamed, or (3) the clients request. By doing this, FairQ should be

able to achieve higher bandwidth an utilization than the resolution-based algorithm.

5.2.1 Quality Smoothing

Research suggests the both frequent and large quality switches contribute negatively to QoE [37, 11]. Quality smoothing is the practice of preventing clients from making quality switches larger than one level at a time. FairQ utilizes quality smoothing when transitioning from the initial phase to the adaptation phase and during the adaptation phase when a client needs to switch quality levels. Quality smoothing decreases the magnitude of quality switches however, it also increases the total number of quality switches and can lower utilization. For this reason, it is a parameter in FairQ that can be turned on or off. We can see the difference that quality smoothing makes in quality levels when transitioning from the initial phase to the adaptation phase in Fig. 5.5.

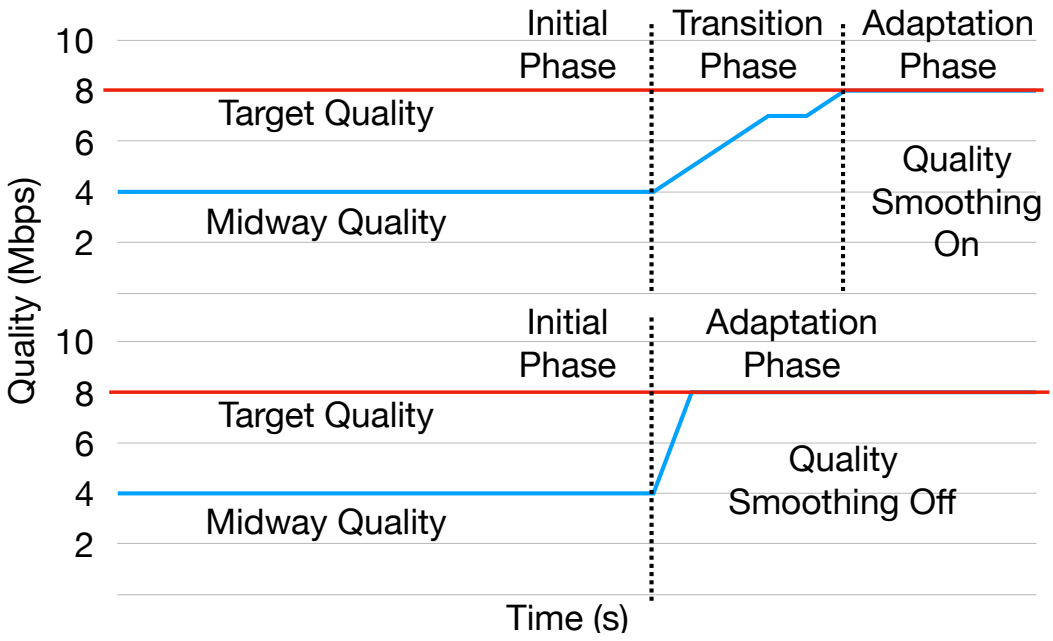

Figure 5.5: The optional transition phase that occurs between the initial phase and adaptation phase when quality smoothing is turned on vs no transition phase when quality smoothing is turned off.

5.2.2 Fairness

Other DASH algorithms focused on fairness, such as FESTIVE, allocate network resources equally amongst all clients to achieve resource fairness. In contrast to this, FairQ attempts to allocate network resources in such a way that achieves equal QoE amongst all clients. This is done by using SSIM, which is a resolution aware QoE measure (we previously explained SSIM in Chapter 3). We have computed the SSIM values of the video "Big Buck Bunny" at three resolutions (360p, 720p, 1080p) as shown in 5.6. The reference video for each resolution is the highest quality video offered at that resolution. To compute the SSIM value of a video that has a different resolution than the reference video, we first upscale the video to the same resolution as the reference video and then compute a SSIM value. It is worth noting that we could use PSNR in place of SSIM, and we have also calculated the PSNR values of the video "Big Buck Bunny" at three resolutions (360p, 720p, 1080p) as shown in 5.7.

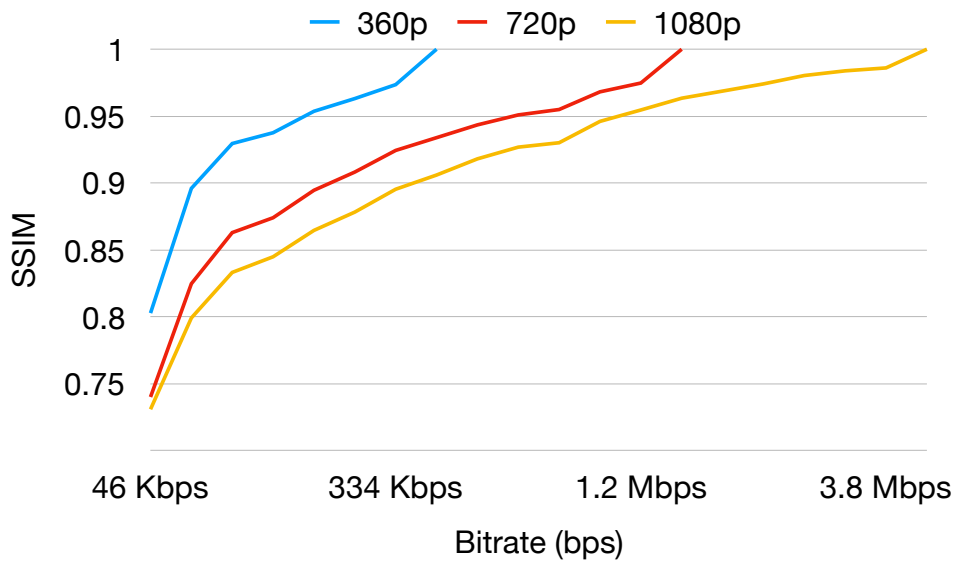

Figure 5.6: SSIM values for the video Big Buck Bunny at 3 resolutions (360p, 720p, 1080p).

In Chapter 4, we used a resolution-based algorithm that simply looped through clients from lowest to highest SSIM value and raised their quality levels until there was not any server bandwidth left or each client was at its maximum quality level. The resolution-based algorithm pro-

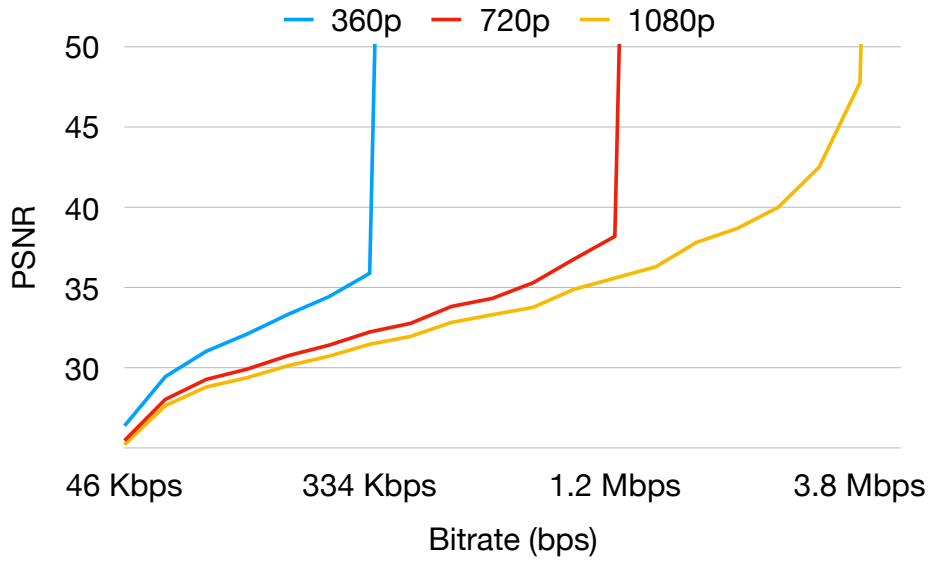

Figure 5.7: PSNR values for the video Big Buck Bunny at 3 resolutions (360p, 720p, 1080p).

duced the best fairness under mixed client scenarios (scenarios where clients have different screen resolutions) however, the use of a constant solution may prevent clients from achieving optimal quality levels depending on the client and network scenarios. FairQ also uses SSIM and client resolutions to achieve QoE fairness however, it does not use a constant solution to determine quality levels. By doing this, FairQ should be able to achieve better fairness than the resolution-based algorithm.

5.3 Design Objectives Revisited

In this section we will show how FairQ achieves each of the designed objectives listed earlier in this Chapter. The design objectives of FairQ are as follows:

- **Objective 1: Produce QoE fairness and not simply resource fairness between clients.** FairQ uses SSIM and client resolutions to achieve QoE fairness like the resolution-based algorithm. Unlike the resolution-based algorithm, FairQ does not use a constant solution to determine quality levels which should allow FairQ to

achieve better fairness than the resolution based algorithm.

- **Objective 2: Produce high bandwidth utilization and quality levels without sacrificing buffer levels.** Like the resolution-based and rate-based algorithms, FairQ will have clients that utilize the harmonic mean of the previous 20 throughput values as bandwidth estimation to achieve high utilization. Using the harmonic mean of the previous 20 throughput values as bandwidth estimation can lead to long and unstable initial phases. To prevent this, FairQ will use the midway quality strategy during the initial phase.

- **Objective 3: Produce a low number of quality switches while still achieving adaptability and maintaining visual quality.** FairQ clients utilize the harmonic mean of the previous 20 throughput values as a bandwidth estimation like the resolution-based algorithm to produce stable quality levels. Unlike the resolution-based algorithm, FairQ wont use a constant solution to determine quality levels that will potentially increase the number of quality switches to achieve greater adaptability.

- **Objective 4: Produce high buffer levels without sacrificing quality levels and utilization.** FairQ only raises the quality level of clients with buffer levels higher than the low buffer thresholds, never raises a clients quality level higher than their request quality, and considers the overhead of DASH to achieve healthy buffer levels. FairQ still allows clients with high requests and high buffer levels to experience high quality levels and thus wont be sacrificing quality levels and utilization.

- **Objective 5: Produce a short and stable initial phase without sacrificing visual quality.** FairQ utilizes the midway quality strategy during the initial phase to achieve a compromise between visual quality and length while also producing a stable initial phase.

5.4 Complexity Analysis

FairQ clients use the harmonic mean of the previous 20 throughput values as a bandwidth estimation that is then used to determine request quality. The formula to calculate the harmonic mean of n numbers (a_1, a_2, ... a_n) is

$$\frac{n}{\sum_n \frac{1}{a_n}} \tag{5.2}$$

This would have a time complexity of $O(n)$ due to the summation however, the time complexity remains constant if n is the number of clients. If we are using 4 byte integers then the space complexity is $O(n)$ where n is the quantity of numbers but would be constant if n is the number of clients.

The pseudo code for the FairQ algorithm is shown in Fig. 5.8, and for each client FairQ stores the clients buffer level, final quality, request quality, and resolution. These are all stored using hash tables due to the constant time insertion, deletion, and search. The loop from lines 2 to 4 in shown in Fig. 5.8 sets the final quality for each client to the minimum of the fair quality level for that clients group and the clients request quality that would have a time complexity of $O(n)$ where n is the number of clients. The loop in lines 7 to 9 shown in Fig. 5.8 calculates the average SSIM value for each resolution group. This involves looking up the final quality level of each client and then looking up the associated SSIM value stored in a hash table for a time complexity of $O(n)$. The loop on lines 10 to 17 involve looping through each client and raising the quality level of higher buffer, high harmonic mean clients. This involves accessing the buffer level, the request quality, and possibly the final quality for all clients for a time complexity of $O(n)$.

The worst case for FairQ is that each client starts at the lowest possible quality level and is raised one level at a time to the largest possible quality level. In that case, the while loop from lines 6 to 20 in Fig. 5.8 would run $q-1$ times where q is the number of quality levels. In the case of the video "Big Buck Bunny", there are 20 quality levels. This means the while loop would run 19 times. Since $q-1$ is a constant value for a given video and all the other loops have a time

complexity of $O(n)$ we can conclude that the FairQ algorithm has a time complexity of $O(n)$. For each client, FairQ stores the clients buffer level, final quality, request quality, and resolution that are all stored using hash tables. FairQ also stores a fair quality level and midway quality level for each resolution group. All of these values are stored using hash tables that have a space complexity of $O(n)$. This means that FairQ utilizes a constant number of hash tables to store all the necessary information making the overall space complexity $O(n)$ where n is the number of clients.

5.5 Implementation

Since FairQ is a client-server solution, it contains adaptation logic on both the client and server side. The server-side adaptation was implemented via a python proxy we developed that intercepts and modifies clients requests before being sent to the server. The design overview of the proxy can be seen in Fig. 5.9. The video server is implemented via Apache 2.0 and also runs on a virtual machine hosted by Cybera at the University of Calgary that has 8GB of ram, 4 virtual CPUs, and 40GB of storage with Ubuntu 14.04 LTS. The video data set is provided by Lederer *et al.* and the video used is called 'Big Buck Bunny' [28]. All bandwidth emulation is done via `tc`, a Linux program for configuring packet scheduling, on the server [25].

As shown in Fig. 5.9, FairQ is a client-server adaptation scheme that contains adaptation logic on both the client and server side. On the client side, FairQ clients utilize the harmonic mean of the previous 20 throughput segments as bandwidth estimation. This was done by modifying the adaption module of the GPAC 0.6.1 player which is an open source media player written in mostly C [27]. The clients are configured via a config file to connect to the FairQ proxy which contains the server-side adaptation. This proxy is implemented in python using standard python libraries and runs on a virtual machine hosted by Cybera at the University of Calgary that has 8GB of ram, 4 virtual CPUs, and 40GB of storage with Ubuntu 14.04 LTS.

As shown in Fig. 5.9, the proxy consists of 3 modules, the main module, the server module, and the adaptation module. The main module obtains information from the configuration file such

Algorithm 5: Pseudo code for FairQ

```
1  B = 93.5% of servers bandwidth ;
2  foreach client i do
3      q_i = min(r_i, f_{q_i}) ;
4  end
5  done = False ;
6  while not done do
7      foreach resolution group do
8          calculate average SSIM value ;
9      end
10     foreach resolution group k from lowest to highest average SSIM do
11         foreach client i in group k do
12             if r_i > q_i and buff_i > buff_low then
13                 raise q_i by one level ;
14             if ∑_i q_i < B then
15                 end algorithm ;
16         end
17     end
18     if no quality levels have been changed then
19         done = True ;
20 end
```

Figure 5.8: Pseudo code for FairQ where r_i, q_i, f_{q_i}, and $buff_i$ are the request quality, quality level, fair quality level, and buffer level of client i and $buff_{low}$ is the low buffer threshold.

as low buffer threshold, server capacity, number of clients, and if quality smoothing is on an off. This information is then passed to the server module which manages all sockets and http requests. The server module creates an adaptation module from the configuration file information. When a client makes a request the server module passes that information to the adaptation module which runs the FairQ algorithm and returns the final quality level to the server module. The server module then modifies the request, sends it to the server, and returns the result to the client.

The FairQ algorithm utilizes SSIM values to determine client quality levels. These SSIM values need to be calculate offline first before the server can be run. We calculated these values using a tool called QPSNR using different resolutions and quality levels of the "Big Buck Bunny" video [38]. To calculate the SSIM of a particular video, a comparison video is needed at the same resolution. To obtain comparison videos of each resolution at each quality level, ffmpeg was used to convert videos to the proper screen resolution [12].

5.5.1 Parameters

The two parameters of FairQ are: (1) the low buffer threshold, and (2) the use of quality smoothing. As shown in Fig. 5.8, the low buffer threshold is used to determine which clients will have their quality levels raised. If the buffer threshold is set too low then clients may experience low buffer levels and possibly stalls in video playback. If the buffer threshold is too high then the bandwidth utilization will be low due to clients not having their quality levels raised. Quality smoothing reduces the magnitude of quality switches at the cost of increasing total quality switches and possibly decreasing bandwidth utilization. When we evaluate FairQ in Chapter 6, we will use multiple low buffer thresholds with both quality smoothing on and off to determine the impact that they have on algorithm performance.

5.5.2 Asynchronous Issues

The main challenge when implementing FairQ were issues involving asynchronous streaming sessions. There are two levels of asynchronicity: phase level and segment level. At the phase level,

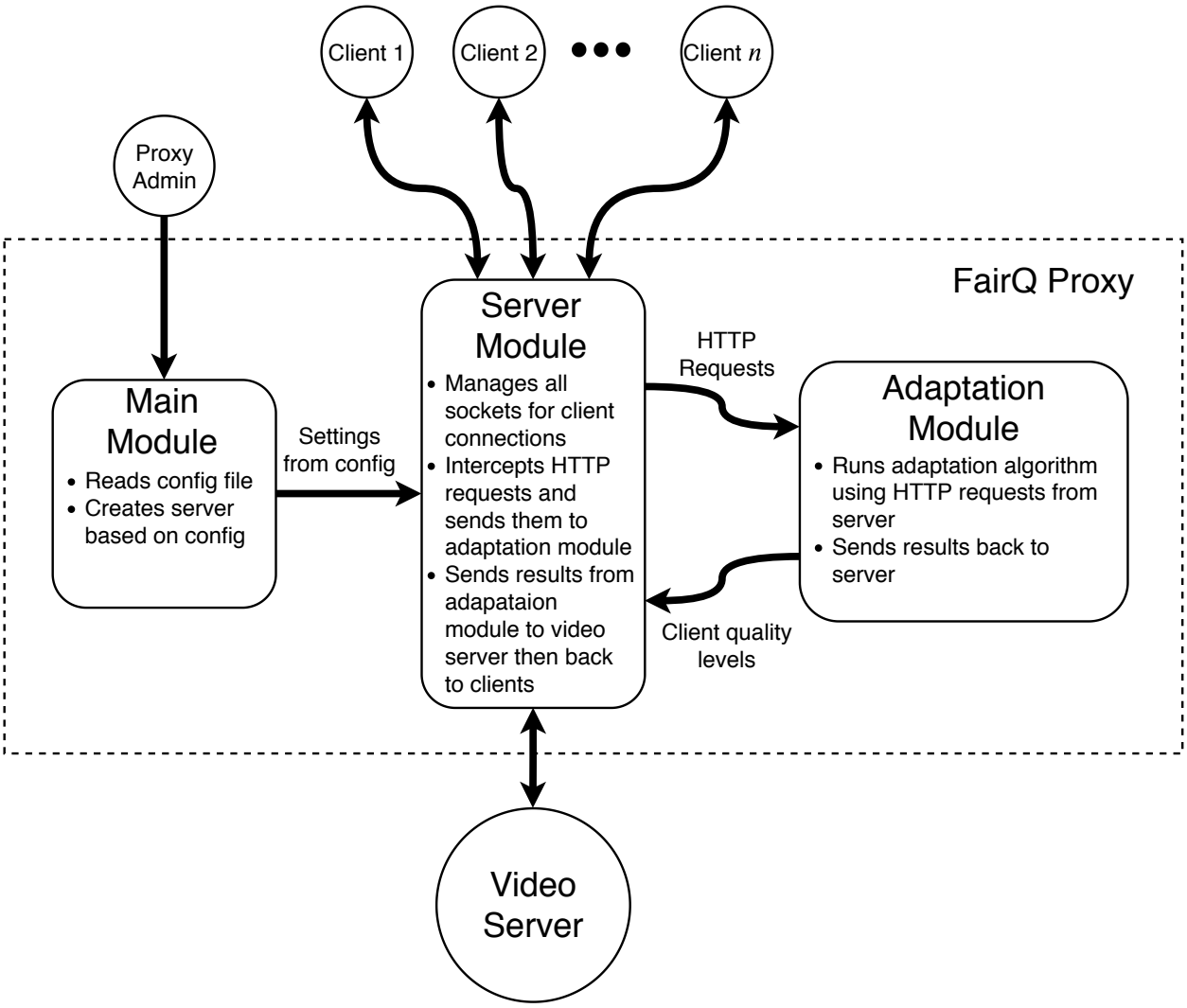

Figure 5.9: An overview of the FairQ algorithm showing the modules and the interaction between the proxy, video server, and clients.

clients could be in any of the three phases (initial, optional transition, and adaptation) at the same time. When a new client joins or leaves, the fair share of the servers bandwidth for each client changes. There are six cases we need to consider: (1) a client joins and one or more clients are in the initial phase. (2) A client joins and one or more clients are in the transition phase. (3) A client joins and one or more clients are in the adaptation phase. (4) A client leaves and one or more clients are in the initial phase. (5) A client leaves and one or more clients are in the transition phase. (6) a client leaves and one or more clients are in the adaptation phase.

To keep the initial phase stable, any clients currently in the initial phase when a client joins or leaves will continue streaming at the same midway quality level for the entirety of the initial phase. This can be seen in Fig. 5.10 where client 2 has a constant midway quality level when client 3 joins and client 2 and 3 both maintain a constant midway quality level when client 1 leaves. To prevent over or under utilization of bandwidth, any clients currently in the adaptation or transition phase will have their quality levels adjusted when a clients joins or leaves. This can be seen in Fig. 5.10 where the quality level of client 1 is lowered when client 2 and 3 join and the client level of client 3 is raised when client 2 leaves.

At the segment level, segment requests do not arrive at the same time. In our implementation, we apply the FairQ adaptation scheme upon receiving each segment to update the quality level. For decodability of segments received on the client side, we cannot change the quality of a segment that is already in transmission. The updated quality level of a segment can be different from the level it is currently being served. FairQ orders client in groups based on their initial arrival time and loops through groups from earliest to latest arrival when raising quality levels. This means that given the same conditions, FairQ will produce the same results regardless of the client with the most recent segment request. This issue can also arise when the fair quality level for each client is updated due to clients joining or leaving or a clients buffer level falls below the low buffer threshold. In either case, any clients with updated quality levels that have segments already in transmission will receive the correct quality segment on subsequent requests. This may affect the

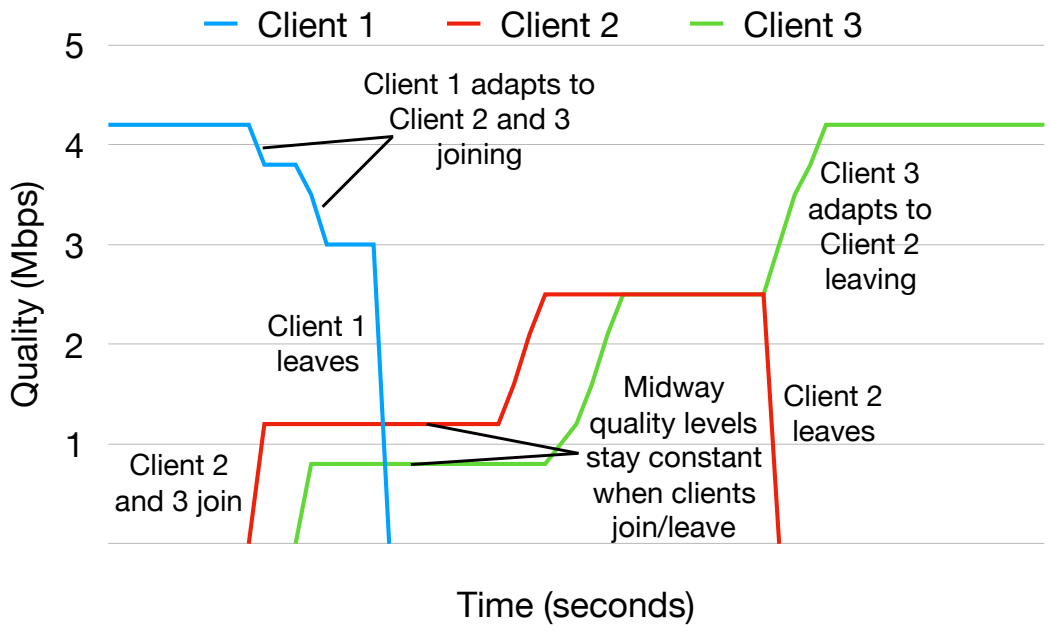

Figure 5.10: Illustrating four cases of phase level asynchronicity.

occasional segment request however, it should not be a large issue that significantly affects the performance of FairQ.

5.6 Summary

In this chapter we presented our client-server adaptation scheme FairQ. During the initial phase, FairQ utilizes the midway quality strategy to achieve a compromise between visual quality and length while also producing a stable initial phase. FairQ utilizes SSIM values and client resolutions to achieve QoE fairness and not simply resource fairness. FairQ clients use the harmonic mean of the previous 20 throughput values as bandwidth estimation to achieve high utilization and stable quality levels. FairQ ensures that clients never receive quality levels greater than the quality level they requested, only raises the quality level of high buffer clients, and considers the overhead of DASH to produce healthy buffer levels without sacrificing utilization or quality levels. FairQ is configurable in terms of low buffer threshold value and whether or not quality smoothing is turned

on or off. Turning quality smoothing on prevents clients from making quality jump larger than one quality level at a time which increases QoE but also increases the total number of quality switches. Turning it off lowers the number of quality switches but allows for large jumps in quality which lowers QoE. The following Chapter will evaluate FairQ under a variety of client and network scenarios using the same comparison algorithms from Chapter 4.

Chapter 6

Evaluation

In Chapter 5, we proposed a client-server solution that controls the quality selection of DASH sessions based on both server bandwidth and client status (including request quality, buffer level, and resolution) called FairQ. FairQ is based on the systematic analysis of DASH algorithms done in Chapters 3 and 4. In this Chapter, we will evaluate FairQ against the results obtained in Chapter 4. The analysis done in Chapter 4 used a rate-based algorithm, buffer-based algorithm, hybrid algorithm, resolution-based algorithm, and FESTIVE. We will begin by describing our experimental setup in section 6.1. In sections 6.2 to 6.8, we will present the results with each section examining one of the 6 metrics used in Chapter 4. In section 6.9, we will examine the overhead of FairQ and section 6.10 will end this chapter with a brief summary.

6.1 Experimental Setup

For each algorithm used in this Chapter, all client side adaptation was implemented in the adaptation module of the GPAC 0.6.1 player. All server side adaptation was implemented in the adaptation module of a python proxy we have developed. The setup is identical to the setup provided in Chapter 3 and 4 with the video server and the proxy each running their own virtual machine hosted by Cybera at the University of Calgary. Each machine has 8GB of ram, 4 virtual CPUs, and 40GB of storage with Ubuntu 14.04 LTS. All clients are running on the same machine that has 8GB of ram, 2 CPUs, 250GB of storage with Ubuntu 14.04 LTS and video playback disabled to minimize CPU usage. The video used is Big Buck Bunny provided by Lederer *et al*. [28]. All bandwidth emulated is done via `tc`, a Linux program for configuring packet scheduling.

Each experiment in this Chapter was repeated 3 times and the results were averaged over the 3 runs. We have assumed that clients cannot request video segments with a higher resolution that

their screen resolution. To achieve this we have simply created two versions of the MPD filed offered by the server, one that includes videos playable by a 1920x1080 client and another that only includes videos playable by a 480x360 client. The servers bandwidth was set to the average of the corresponding network scenario when the tests were run.

To evaluate FairQ, we will select the best algorithm for each metric from Chapter 4 and compare that against the performance of FairQ. When the results are presented, the best value for each metric under each scenario will be listed in a row labelled "Best" and the algorithm that contributed that value will be listed in the row below. The metrics used are as follows:

- **Fairness**: The standard deviation of average SSIM between clients over all runs.
- **Utilization**: The average bandwidth utilization over all runs, listed at "Util." in the table.
- **Average Number of Quality Switches**: The average number of quality switches for all clients over all runs, listed as "Switches" in the table.
- **Average Buffer Levels**: The average buffer level for all clients over all runs in seconds, listed as "Buffer" in the table.
- **Initial Phase Length**: The average initial phase length for all clients over all runs in seconds, listed as "I.P. Length" in the table.
- **Average Segment Quality**: The average quality of segments for all clients over all runs in Mbps, listed as "Quality" in the table.

6.2 Default Settings

As shown in Table 6.1, we will only be using the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios. In Chapter 4, the results showed that different network scenarios had little impact on tests results, with the exception of the Real Bandwidth Trace. For each variation of

Network Scenarios	Drop 1, Big Drop, Real Bandwidth Trace
Client Scenarios	Homogeneous, Mixed
Number of Clients	4, 20
FairQ Quality Smoothing	off
FairQ Buffer Thresholds	20s (homogeneous), 25s (mixed)

Table 6.1: Default settings used during this Chapter that include network scenarios, client scenarios, number of clients, and FairQ parameters.

FairQ, we have created six tables, shown in Appendix B, for each metric used during evaluation. Each of these tables show the performance of that variation of FairQ across network scenarios and can be seen in Tables B.1 to B.36 in Appenix B. FairQ **with** quality smoothing is listed as 'FairQ-smooth' and **without** is simply 'FairQ'. These tables show that the performance under each metric is very similar across all network scenarios with the exception of Fairness. For fairness, there are two groups created with Drops 1 and 2 and the Drop and Raise scenario having similar results and all the other scenarios producing similar results to each other shown in Tables B.1 to B.6.

For this reason and for the sake of brevity, we will only present the results from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios. These network scenarios can be seen in Fig. 6.1. Drop 1 switches between bandwidth limits of 9.5 and 11 Mbps with a frequency of 30 seconds. Big Drop switches between bandwidth limits of 8, 9, and 11 Mbps with a frequency of 30 seconds. A bandwidth limit of 11 Mbps should allow each client to stream at a quality of 2.5 Mbps under homogeneous client scenarios. A bandwidth limit of 9.5 Mbps should allow each client to stream at a quality of 2.1 Mbps under homogeneous client scenarios. A bandwidth limit of 8 Mbps should allow each client to stream at a quality of 1.5 Mbps under homogeneous client scenarios. A bandwidth limit of 11 Mbps and 9.5 Mbps should allow both groups to stream at their maximum quality levels of 4.2 Mbps and 396 Kbps respectively under mixed client scenarios. A bandwidth limit of 8 Mbps should allow 1920x1080 clients to stream at a quality of 3.5 Mbps and 480x360 clients to stream at a quality level of 396 Kbps under mixed client scenarios. Both of these network scenarios will be used with 4 clients to get a fair comparison with the algorithms from Chapter 4.

The dataset for the Real Bandwidth Trace was provided by Raca *et al.* and contains throughput

values obtained over a 4G cellular network from a single client at a granularity of one sample per second [45]. The Real Bandwidth Trace will be tested using 20 clients, thus, we have simply multiplied the trace by 20 to achieve the bandwidth limit shown in Fig. 6.1. Again, 20 clients will be used to get a fair comparison with the algorithms in Chapter 4.

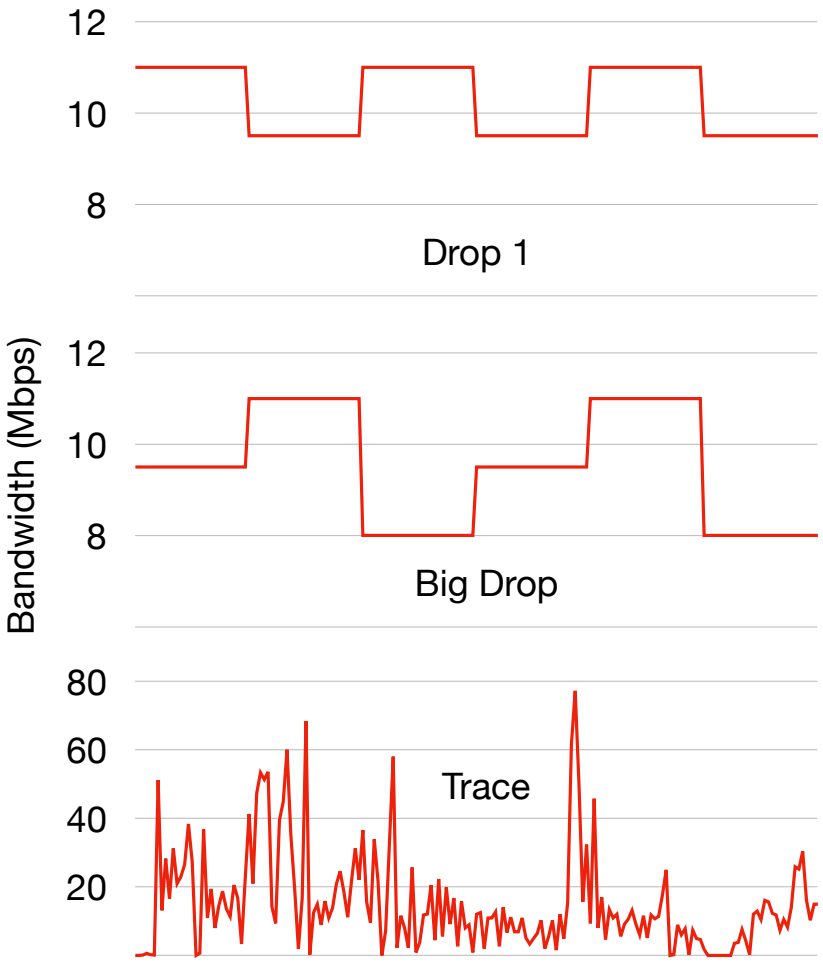

Figure 6.1: The network scenarios that will be analyzed during this Chapter.

Each network scenario will be tested using both a homogeneous and a mixed client scenarios except for the Real Bandwidth Trace, which will only use a mixed client scenario. This is done to get a fair comparison with the algorithms in Chapter 4. The homogeneous client scenario consists

of exclusively clients with a resolution of 1920x1080 and the mixed client scenario consists of half clients with a resolution of 1920x1080 and the other half with a resolution of 480x360. We will also re-analyze the results of the homogeneous clients scenarios and obtain metrics for just 1080p clients and just 360p clients.

We tested a total of six different versions of FairQ that consist of Fair with quality smoothing, and FairQ without quality smoothing. Each variation was tested using buffer thresholds of 15, 20, and 25 seconds. For the sake of brevity, we will pick the best variation of FairQ under homogeneous client scenarios and the best variation of FairQ under mixed client scenarios and only present the results from those variations. The results in their entirety can be seen in Appendix A. To determine the best variation under each client scenario, we will simply pick the best variation using the Drop 1 network scenario.

As shown in Table 6.2, the best variation under the homogeneous client scenario is FairQ with a 20 second buffer threshold due to achieving the best performance in 4 out of the 6 metrics. As shown in Table 6.3, the best variation under the mixed client scenario is FairQ with a 25 second buffer threshold due to achieving the best performance in 4 out of the 6 metrics. Since the best variation of FairQ under both client scenarios is FairQ without quality smoothing, our default setting is that quality smoothing is turned off as shown in Table 6.1. Similarly, the low buffer thresholds for the best variations of FairQ are 20 and 25 seconds for homogeneous and mixed client scenarios respectively. This is why our default setting only includes the low buffer thresholds of 20 and 25 seconds as shown in Table 6.1. In the following sections, we will examine the results with a focus on each metric and the performance of FairQ against the top performer from Chapter 4. We will simply use “FairQ” in place of “FairQ 20s” or “Fair2 25s” when presenting the results.

6.3 Fairness

In this section, we will examine the performance of FairQ focusing on Fairness. As shown in Table 6.4, FairQ produced the best fairness overall under all mixed client scenarios with an increase of

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
FairQ-smooth 15s	0.001616	79.4	4.5	**28.2**	13.4	2.083
FairQ 15s	**0.001370**	80.3	3.0	27.9	13.4	2.108
FairQ-smooth 20s	0.002102	79.3	4.3	**28.2**	13.5	2.081
FairQ 20s	0.001543	**80.4**	**2.7**	27.9	**13.2**	**2.109**
FairQ-smooth 25s	0.001579	79.3	4.7	**28.2**	13.6	2.082
FairQ 25s	0.001970	80.1	2.9	28.0	13.5	2.103

Table 6.2: Results from the Drop 1 network scenario under a homogeneous client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
FairQ-smooth 15s	0.002470	79.7	4.5	**28.7**	9.7	2.092
FairQ 15s	0.002194	80.7	**1.0**	28.6	9.6	2.118
FairQ-smooth 20s	0.002562	80.0	4.5	**28.7**	**9.4**	2.099
FairQ 20s	**0.002136**	80.7	**1.0**	28.5	9.8	2.118
FairQ-smooth 25s	0.002446	79.9	4.5	**28.7**	9.5	2.098
FairQ 25s	0.002235	**80.8**	**1.0**	28.4	**9.4**	**2.120**

Table 6.3: Results from the Drop 1 network scenario under a mixed client scenario.

65%, 48%, and 47% over the resolution-based algorithm under Drop 1, Big Drop, and Real Bandwidth Trace network scenarios. Like the resolution-based algorithm, FairQ uses SSIM values and client resolutions to achieve QoE fairness however, by not using a constant solution to determine quality levels, FairQ achieves greater adaptability resulting in greater fairness overall. FESTIVE tends to produce the best fairness under homogeneous client scenarios due to its long ramp-up behavior that produces nearly identical quality levels. FairQ is generally unable to compete with FESTIVE in homogeneous client scenarios in term of fairness with the exception of the Big Drop network scenario. The Big Drop network scenario has a large bandwidth drop 60 seconds into the streaming sessions that disrupts the ramp-up behavior of FESTIVE and lowers the fairness it achieves. This is why FairQ was able to achieve the best fairness under the Big Drop network scenario.

As shown in Table 6.4, FairQ produced the best fairness for 1080p clients under mixed client scenarios with the exception of the Big Drop network scenario. This is because the Big Drop network scenario has a lower average bandwidth than Drop 1 and FairQ considers the overhead

of DASH. Under Drop 1, FairQ is able to provide both 1080p clients with the maximum quality level of 4.2 Mbps. Under the Big Drop, FairQ is only able to provide one 1080p client with the maximum quality of 4.2 Mbps while the other 1080p clients gets a quality of 3.8 Mbps. This issues does not affect the resolution-based algorithm because it doesn't consider the overhead of DASH. This can be seen in Fig. 6.2, where each FairQ client has a different quality level and each resolution-based client has a nearly identical quality level.

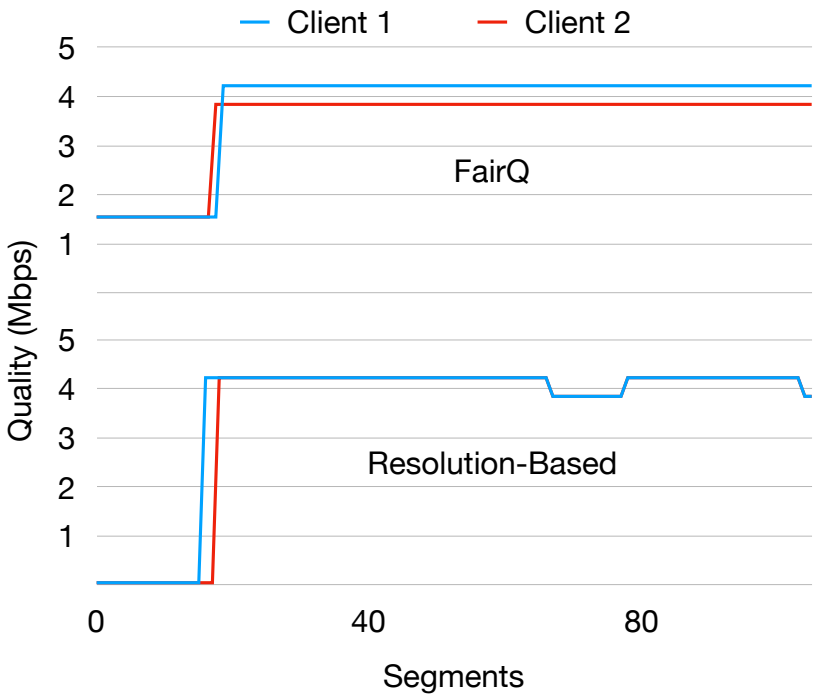

Figure 6.2: Quality levels of 1080p FairQ clients and 1080p resolution-based clients under the Big Drop network scenario.

As shown in Table 6.4, FairQ failed to achieve the best fairness for 360p clients under mixed client scenarios. Under the Real Bandwidth Trace, this is due to high utilization of 1080p clients that reduces the available bandwidth for 360p clients. This leads to a large variation of quality level for 360p clients and ultimately low fairness. This is in contrast to the buffer-based algorithm that produced low utilization for 1080p clients that allows 360p clients to maintain stable quality levels and achieve the best fairness. This will be explained further when we analyze utilization.

Under Drop 1 and the Big Drop, the quality levels of the 360p clients between FairQ and the resolution-based algorithm are identical with the exception of the initial phase quality level. This can be seen by comparing the quality levels of FairQ in Fig. 6.3 and the resolution-based algorithm in Fig. 6.3 under Drop 1. This doesn't account for the difference in fairness. The difference in fairness is due to the number of segments each client requests and how fairness is calculated. We calculate fairness during a session by converting the quality level of each segment requested by a client to a SSIM value using Table 6.5. We then average all the SSIM values to get an average SSIM for each client. The fairness of the session is the standard deviation of the average SSIM values of each client in the session. When we examined the quality levels of 360p resolution-based client, we noticed that each client without exception requested a total of 106 segments during the streaming session. In contrast to this, 360p FairQ clients range from requesting 104 to 106 segments during a streaming session. This difference in number of segments requested results in a higher standard deviation of average SSIM values leading to worse fairness. This is likely due to the overhead introduced by the FairQ proxy that is not present with the resolution-based algorithm because the algorithm is so simple. This difference can be seen in Fig. 6.3 where client 2 requests 1 more segment than client 1 for the FairQ algorithm.

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	0.001543	**0.001970**	**0.000000**	0.000114
Best	**0.000438**	0.005653	0.00007297	**0.00000000**
	FESTIVE	resolution-based	FESTIVE	resolution-based
Big Drop				
FairQ	**0.000407**	**0.003639**	0.00509816	0.00005275
Best	0.000519	0.006973	**0.00040102**	**0.00000000**
	FESTIVE	resolution-based	resolution-based	resolution-based
Trace				
FairQ		**0.007739**	**0.001038**	0.002301
Best		0.014651	0.001368	**0.000129**
		resolution-based	FESTIVE	buffer-based

Table 6.4: Fairness results from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

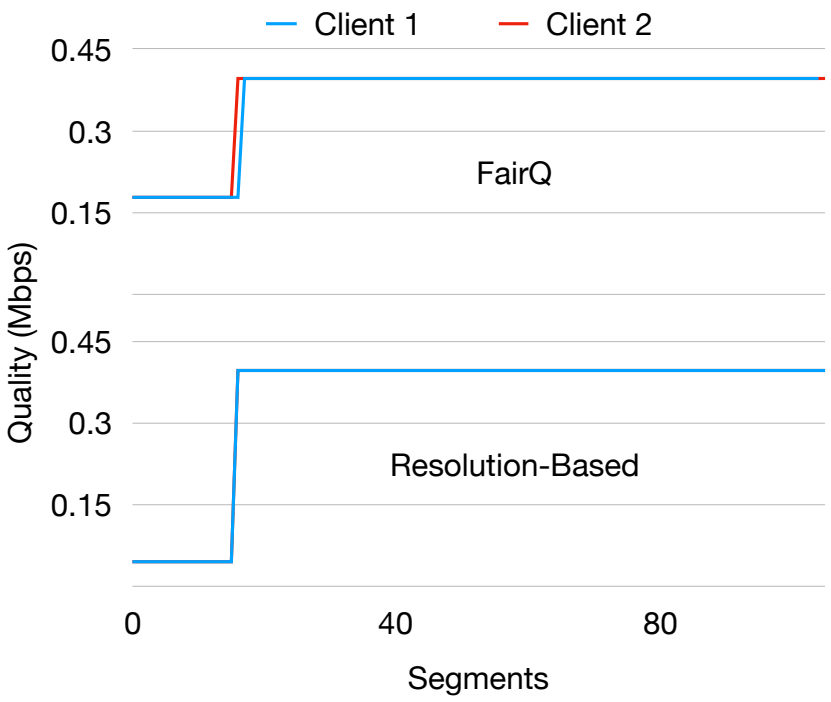

Figure 6.3: Quality levels of 360p FairQ clients and 360p resolution-based clients under the Drop 1 network scenario.

6.4 Utilization

In this section, we will examine the performance of FairQ focusing on Utilization. As shown in Table 6.6, FairQ achieved the highest utilization under mixed client scenarios with an improvement of 8% and 3% for the Drop 1 and Big Drop network scenarios over the best algorithms from Chapter 4. In both cases, the best algorithm from Chapter 4 is the resolution-based algorithm which requests the lowest quality segments during the initial phase and places an upper limit on client quality levels. By using the midway quality strategy during the initial phase and not placing an upper limit on client quality levels, FairQ is able to achieve higher utilization. As shown in Table 6.6, FairQ failed to achieve the highest utilization in the Real Bandwidth Trace network scenario under a mixed client scenario however, it did achieve the second highest utilization. The rate-based algorithm achieved the highest utilization due to high quality levels during the initial phase at the cost of a long, unstable initial phase and low buffer levels. Thus, we consider the trade off between shorter, more stable initial phases and better buffer levels and 2% lower utilization to be justified.

As shown in Table 6.6, FairQ achieved the highest utilization for 1080p clients under mixed client scenarios with an increase of 8%, 3%, and 14% for the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios respectively. However, FairQ failed to achieve the highest utilization for 360p clients under mixed client scenarios across network scenarios. This is a result of FairQ achieving the the highest utilization for 1080p clients. By having the highest utilization, 1080p FairQ clients leave less bandwidth for 360p clients and increase the chance that segment downloads will overlap leading to low throughput values. This is because FairQ picks the group with the lowest average SSIM and raises the quality levels of all eligible clients in that group before moving on to the next group. Looking at Table 6.5, we can see that 1080p clients have more quality levels to choose from and the increase in SSIM is much less between adjacent quality levels compared to 360p clients. This means FairQ tends to focus more on 1080p clients as raising the quality level of a 360p would have a greater impact on the groups average SSIM than raising the

quality level of a 1080p client. We don't consider the lower utilization of 360p clients to a be a big issue as FairQ still achieved the best fairness and second highest utilization overall.

		SSIM		
Quality	**Resolution**	**360p**	**720p**	**1080p**
178 Kbps	480x360	0.937617	0.874103	0.844944
222 Kbps	480x360	0.953646	0.894626	0.864652
263 Kbps	480x360	0.963134	0.908238	0.878324
334 Kbps	480x360	0.973549	0.924399	0.895398
396 Kbps	480x360	1	0.933925	0.906005
522 Kbps	854x480		0.943478	0.918096
595 Kbps	854x480		0.950922	0.926849
791 Kbps	1280x720		0.954965	0.93014
1 Mbps	1280x720		0.968168	0.946091
1.2 Mbps	1280x720		0.974747	0.95461
1.5 Mbps	1280x720		1	0.963415
2.1 Mbps	1920x1080			0.968729
2.5 Mbps	1920x1080			0.97408
3.1 Mbps	1920x1080			0.98036
3.5 Mbps	1920x1080			0.983855
3.8 Mbps	1920x1080			0.986035
4.2 Mbps	1920x1080			1

Table 6.5: SSIM values for the "Big Buck Bunny" video at 3 resolutions (360p, 720p, 1080p).

6.5 Quality

In this section, we will examine the performance of FairQ focusing on Quality. As shown in Table 6.7, the same trends from the Utilization analysis are present due to the fact that quality levels and bandwidth utilization are so closely linked. For the sake of brevity we wont repeat the analysis here however, we will provide a brief summary. FairQ achieved the highest quality levels under mixed client scenarios with an improvement of 8% and 3% for the Drop 1 and Big Drop network scenarios over the best algorithms from Chapter 4 due to the use of the midway quality strategy and not limiting client quality levels. FairQ failed to achieve the highest overall quality levels under the Real Bandwidth Trace by achieving a shorter, more stable initial phase and higher buffer levels than the rate-based algorithm. FairQ failed to achieved the highest quality levels for 360p clients

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	80.4	**80.8**	**73.9**	6.9
Best	**91.9**	74.8	68.2	**7.5**
	rate-based	resolution-based	resolution-based	rate-based
Big Drop				
FairQ	81.2	**82.1**	**74.8**	7.3
Best	**93.6**	79.6	72.5	**8.2**
	rate-based	resolution-based	resolution-based	rate-based
Trace				
FairQ		80.8	**67.6**	13.2
Best		**82.8**	59.1	**25.9**
		rate-based	resolution-based	buffer-based

Table 6.6: Utilization (%) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

under mixed client scenarios due to focusing on 1080p clients because they have more possible SSIM values as shown in Table 6.5. This ultimately leads to 1080p clients having the highest quality levels leaving less available bandwidth for 360p clients.

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	2.109	**2.120**	**3.878**	0.363
Best	**2.411**	1.963	3.583	**0.395**
	rate-based	resolution-based	resolution-based	rate-based
Big Drop				
FairQ	1.961	**1.984**	**3.614**	0.354
Best	**2.263**	1.923	3.504	**0.395**
	rate-based	resolution-based	resolution-based	rate-based
Trace				
FairQ		0.613	**1.026**	0.200
Best		**0.628**	0.897	**0.393**
		rate-based	resolution-based	buffer-based

Table 6.7: Quality levels (Mbps) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

6.6 Switches

In this section, we will examine the performance of FairQ focusing on Quality Switches. As shown in Table 6.8, FairQ produced the lowest number of switches across all client scenarios under the Drop 1 and Big Drop network scenarios with an average decrease of 34% over the best algorithms from Chapter 4. We attribute this to clients that use the harmonic mean of the previous 20 throughput values as bandwidth estimation, considering the overhead of DASH that prevents clients from over estimating bandwidth, and the fact that quality smoothing is turned off. As shown in Table 6.8, FairQ failed to produce the lowest number of switches under the Real Bandwidth Trace across all client scenarios. FairQ still achieved the second lowest number of quality switches for 1080p clients however, it achieved the highest number of quality switches for 360p clients. The slight increase in number of quality switches for the 1080p clients over the resolution-based algorithm is a result of not placing an upper limit on client quality levels. This can be seen by comparing the quality levels of a 1080p FairQ, shown in Fig. 6.5, and a 1080p resolution-based client, shown in Fig. 6.6. The 1080p resolution-based client never rises above the limit of 1.24 Mbps which decreases the total number of quality switches at the cost of utilization.

For 360p clients, the high number of switches is a result of the use of a low buffer threshold, the lack of a constant solution, and the use of the harmonic mean of the previous 20 throughput values as a bandwidth estimation. Like the rate-based and resolution-based algorithms, FairQ clients use the harmonic mean of the previous 20 throughput values as bandwidth estimation. This tends to produce stable quality levels however, this bandwidth estimation isn't very reactive to sudden network changes and can result in unstable buffer levels as a result. This can be seen in Fig. 6.4 that shows the buffer levels of a 1080p rate-based client.

FairQ clients experience similar buffer levels however, the FairQ algorithm would stop raising the quality level of a client when their buffer levels fell below the low buffer thresholds and raise the quality level of clients with higher buffer levels. If we look at the buffer and quality levels of a 360p and 1080p FairQ client in Fig. 6.5, we can see that drop in buffer levels around 100 seconds

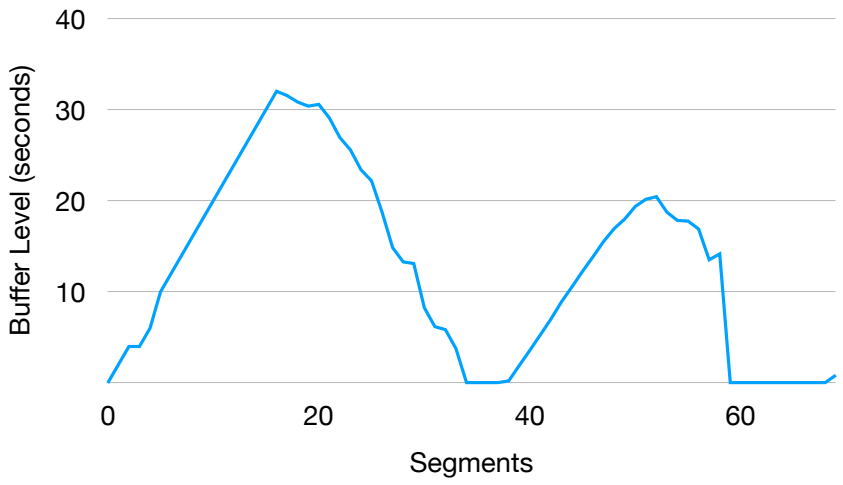

Figure 6.4: Buffer levels of a rate-based 1080p client under the Real Bandwidth Trace.

of the 1080p client result in the quality levels being lowered for the 1080p client and raised for the 360p client. This increases the total number of quality switches. We can also see that the quality levels are very unstable directly after the initial phase. This is a result of FairQ attempting to use all the available bandwidth because other clients are still in the initial phase making more bandwidth available. This is an issue of FairQ that we will discuss in the next Chapter. This doesn't affect the resolution-based algorithm because of the use of a constant solution that limits the maximum quality levels of all clients. This can be seen in Fig. 6.6 that shows the quality levels of a 360p and 1080p resolution-based client during the Real Bandwidth Trace.

6.7 Buffer

In this section, we will examine the performance of FairQ focusing on Buffer Levels. The results in this section show that FESTIVE produced the highest buffer levels across nearly all scenarios due to having very low bandwidth utilization. To get a better idea of how FairQ compares to the algorithms used in Chapter 4, we have introduced the second best algorithm under each client scenario to the analysis. As shown in Table 6.9, FairQ produced the second best buffer levels in 6 of the 11 scenarios and the best buffer levels under 1 scenario. We attribute this to the fact that

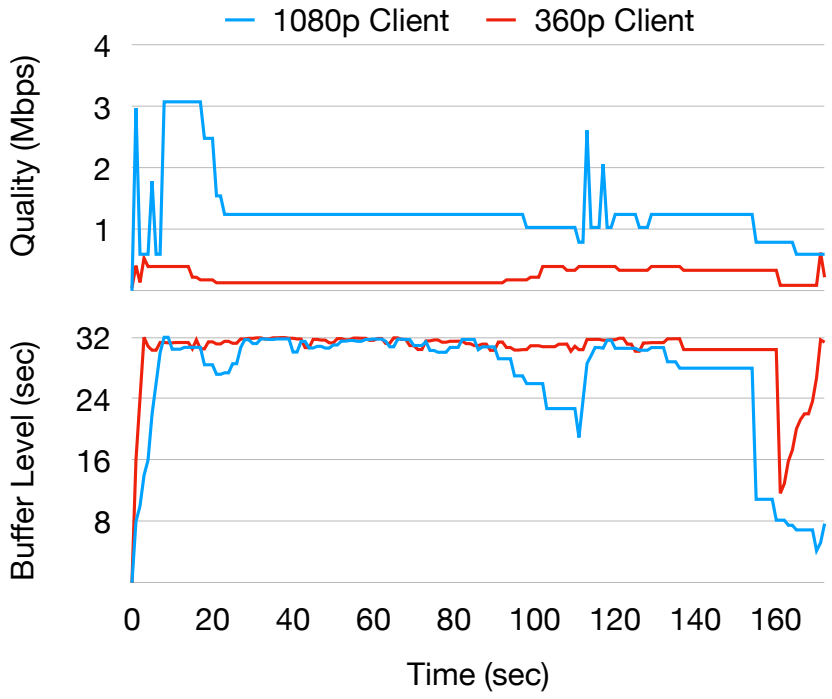

Figure 6.5: Buffer and Quality levels over time of a 360p and 1080p FairQ client during the Real Bandwidth Trace.

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	**2.7**	**1.0**	**1.0**	**1.0**
Best	4.3	1.2	1.3	1.0
	resolution-based	resolution-based	resolution-based	resolution-based
Big Drop				
FairQ	**2.1**	**1.5**	**2.0**	**1.0**
Best	3.3	2.5	4.0	1.0
	resolution-based	resolution-based	resolution-based	resolution-based
Trace				
FairQ		13.2	9.7	16.8
Best		**4.5**	**7.9**	**1.2**
		resolution-based	resolution-based	resolution-based

Table 6.8: Number of quality switches from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

Figure 6.6: Quality levels of a 360p and 1080p resolution-based client under the Real Bandwidth Trace.

FairQ considers the overhead of DASH, only raises the quality level of high buffer clients, and never sends a client a quality level higher than they requested. FairQ failed to produce the highest buffer level in all but one scenario however, there is no benefit in sacrificing quality levels and utilization to increase buffer levels. QoE only suffers when buffer levels hit 0 and causes stalls in video playback. Thus, we consider the buffer levels produced by FairQ to be completely adequate.

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	27.9	28.4	27.9	28.9
Best	**29.0**	**29.4**	**29.2**	**29.6**
2nd Best	27.3	28.6	28.3	29.5
	resolution-based	resolution-based	resolution-based	resolution-based
Big Drop				
FairQ	28.2	28.5	28.1	28.9
Best	**28.9**	**29.3**	**29.2**	**29.7**
2nd Best	27.7	28.5	28.1	29.5
	resolution-based	resolution-based	resolution-based	resolution-based
Trace				
FairQ		26.2	24.5	**28.0**
Best		**26.9**	**26.4**	27.9
2nd Best		25.8	23.7	27.9
		resolution-based	resolution-based	buffer-based

Table 6.9: Buffer level (seconds) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

6.8 Initial Phase Length

In this section, we will examine the performance of FairQ focusing on Initial Phase Length. As shown in Table 6.10, FairQ failed to produce the shortest initial phase across all scenarios. This is due to the fact that all the top algorithms request the lowest quality segments during the initial phase. This results in a short and stable initial phase at the cost of visual quality. FairQ uses the midway quality strategy during the initial phase that produces longer initial phase lengths but also higher visual quality during the initial phase. We consider this a justified compromise and are satisfied with the initial phase performance of FairQ. The difference in quality levels during the

initial phase can be seen in Fig. 6.7 where the resolution-based client requests the lowest quality segments during the initial phase and the FairQ client requests midway quality segments.

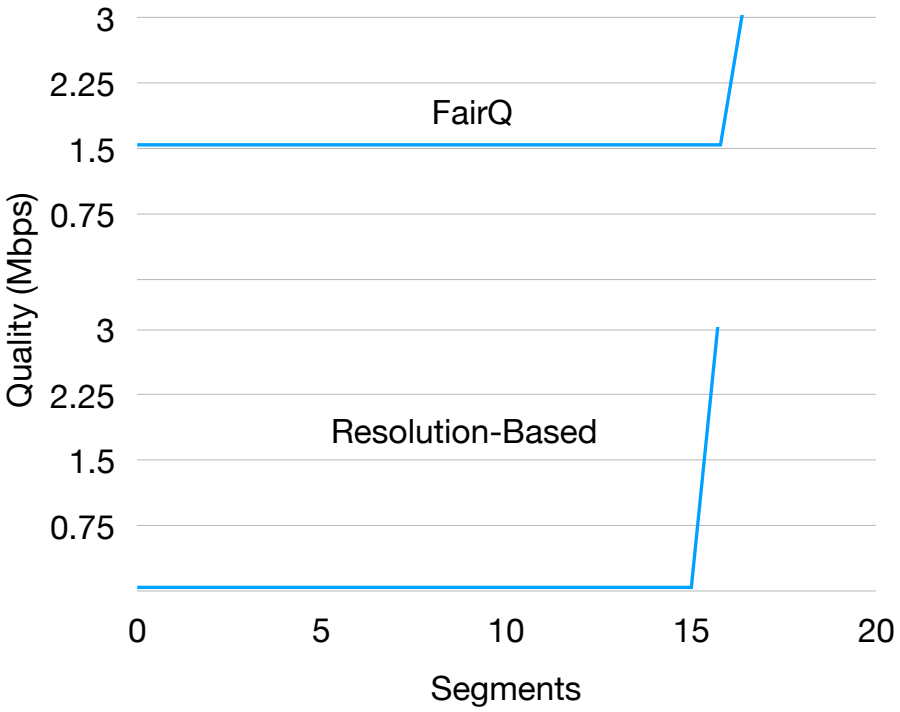

Figure 6.7: Quality levels during the initial phase of a FairQ client compared to a resolution-based client.

6.9 Overhead of FairQ

To determine the overhead of FairQ we measured the CPU usage during a test run using the real bandwidth trace network scenario and 20 clients under the mixed client scenario. We measured the CPU usage using the linux tool `top` at a granularity of 1 second. `top` determines CPU usage by measuring a task's share of the elapsed CPU time since the last update to a get a % of CPU usage. As shown in Fig. 6.8, the CPU usage of FairQ ranges from 0 to 46.8% with an average of 15.5%. While the percentage of time spent idle ranges from 87.6% to 100% with an average of 95.9%. We are satisfied with this performance due to the high idle percentage however, the efficiency of FairQ

Drop 1	**Homogeneous**	**Mixed**		
		Overall	**1080p Clients**	**360p Clients**
FairQ	13.2	9.4	16.1	2.8
Best	**0.6**	**0.6**	**0.6**	**0.5**
	resolution-based	resolution-based	resolution-based	hybrid
Big Drop				
FairQ	15.9	8.5	14.4	2.6
Best	**0.7**	**0.7**	**0.9**	**0.6**
	resolution-based	hybrid	resolution-based	resolution-based
Trace				
FairQ		7.6	13.4	1.8
Best		**1.2**	**0.9**	**1.1**
		FESTIVE	FESTIVE	hybrid

Table 6.10: Initial phase length (seconds) from the Drop 1, Big Drop, and Real Bandwidth Trace network scenarios showing the performance of FairQ and the best algorithm for each metric.

may need to be examined more closely during future work.

6.10 Summary

In this Chapter, we evaluated the performance of FairQ using the following metrics: Fairness, Utilization, Quality Levels, Quality Switches, Buffer Levels, and Initial Phase Length. We compared the performance of FairQ to the performance of the best algorithm from Chapter 4 under each metric. We found the following: (1) FairQ produced the best fairness under mixed client scenarios with an average increase of 53% over the best algorithms from Chapter 4. (2) FairQ produced the best utilization and quality levels for mixed client scenarios under Drop 1 and Big Drop with an average increase of 6% over the best algorithms from Chapter 4. FairQ also produced the second best utilization and quality levels under the Real Bandwidth Trace with a decrease of only 2% over the best algorithm from Chapter 4. (3) FairQ produced the lowest number of quality switches under Drop 1 and Big Drop network scenarios across all client scenarios with an average decrease of 34% over the best algorithms from Chapter 4. (4) FairQ produced adequate buffer levels often having the second highest buffer level compared to algorithms from Chapter 4. (5) FairQ failed to produce the shortest initial phases however, this is due to using the midway quality strategy that

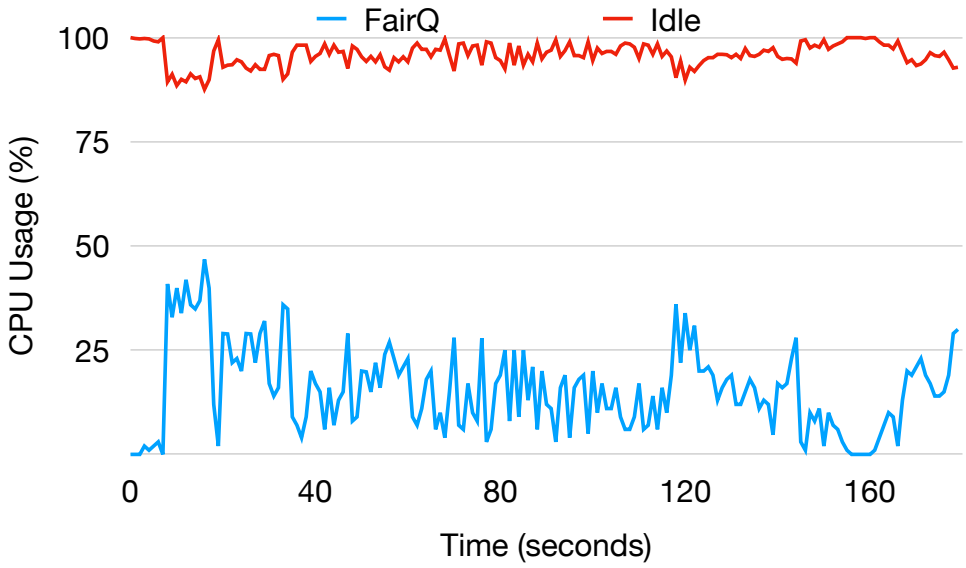

Figure 6.8: CPU usage of FairQ on the proxy during a run using the real bandwidth trace and 20 clients.

results in stable initial phases with higher visual quality.

Chapter 7

Conclusion and Future Work

In this thesis we presented FairQ, a client-server adaptation scheme that controls the quality selection of DASH sessions based on both server bandwidth and client status (including request quality, buffer level, and resolution). FairQ balance the inter-session fairness and intra-session QoE by considering both bandwidth allocation across sessions and appropriate quality adaptation within individual sessions. Through several experiments, we demonstrated the strengths and weakness of FairQ. The experiments using mixed client scenarios (scenarios that include clients of different resolutions), showed that FairQ was able to produce the highest bandwidth utilization, highest quality levels, and highest degree of fairness. FairQ also performed well under a real bandwidth trace producing the highest degree of fairness and near optimal utilization and buffer levels. Given the growth of video streaming and the wide array of heterogeneous clients from smartphone to 4K television, we believe FairQ is an effective way to provide a high QoE for these clients while also providing fairness between them. More experiments can be done using more real bandwidth traces, a larger number of clients, a larger diversity of client resolutions, more variations of FairQ, and a larger variety of comparison algorithms.

7.1 Thesis Summary

In Chapter 1, we explained the motivation of this thesis and introduced adaptive streaming techniques as a way to provide video to a large variety of heterogeneous clients. We briefly discussed typical adaptive streaming approaches that seek to achieve QoE within streaming sessions and achieve fairness across multiple streaming sessions. Further, we stated the objectives of our research and discussed our most significant contributions. We concluded by explaining our plan for the remainder of the thesis.

In Chapter 2, we presented an overview of DASH techniques by grouping algorithms into several broad categories that include: (1) rate-based algorithms, (2) buffer-based algorithms, (3) hybrid of rate-based and buffer-based algorithms, (4) resource fairness algorithms, and (5) QoE fairness algorithms. Rate-based algorithms utilize throughput values to determine segment qualities. Buffer-based algorithms utilize buffer levels and the rate of change of buffer levels to determine segment qualities. Hybrid algorithms utilize both throughput values and buffer levels to determine segment qualities. Resource fairness algorithms attempt to achieve fairness by providing clients with equal network resources while QoE fairness algorithms attempt to achieve fairness by providing clients equal QoE which may or may not lead to equal network resources. We then presented psuedo-code for a rate-based, buffer-based, hybrid, resource fairness, and QoE fairness algorithm that were used throughout this thesis.

In Chapter 3, we performed an analysis of fairness and QoE in DASH using the rate-based, buffer-based, hybrid, resource fairness, and QoE fairness algorithms shown in Chapter 2. We began by introducing out experimental setup and performed tests to validate it. We then determined that the minimum bandwidth needed to stream at a quality level is on average 6.5% higher than the average bitrate listed in the MPD that we refer to as the overhead of DASH. We compared the performance of the rate-based, buffer-based, and hybrid algorithms and presented the strengths and weaknesses of each algorithm. We then presented our own midway quality technique during the initial phase that provided a compromise between visual quality and visual stability. Next, we showed fairness issues that arise when multiple clients are streaming simultaneously and presented client-side and server-side solutions to achieve resource fairness. Finally, we presented a QoE fairness algorithm, refered to as the resolution-based algorithm, and showed the effectiveness of using clients resolutions and SSIM values to achieve QoE fairness.

In Chapter 4, we performed a comparison of adaptation algorithms under a variety of network and client scenarios. We based our choice of algorithms on the results of Chapter 3 and tested them under a variety of dynamic network scenarios, including a real bandwidth trace, using homoge-

neous and mixed client scenarios. We explained the ways in which different algorithms interacted with the dynamic network scenarios that lead to favorable and less favorable results. Under the homogeneous client scenarios, each algorithm performed well under a one or two metrics but not so well under others resulting in no clear algorithm being the best overall. Under the mixed client scenarios, the resolution-based algorithm was the clear winner showing the effectiveness of server-side solutions that utilize clients resolutions and SSIM values to produce QoE within sessions and QoE fairness across sessions.

In Chapter 5, we presented the design of FairQ, our server-client adaptation scheme based on the results of Chapter 3 and Chapter 4. During the initial phase, FairQ utilizes the midway quality strategy to achieve a compromise between visual quality and length while also producing a stable initial phase. FairQ utilizes SSIM values and client resolutions to achieve QoE fairness and not simply resource fairness. FairQ clients use the harmonic mean of the previous 20 throughput values as bandwidth estimation to achieve high utilization and stable quality levels. FairQ ensures that clients never receive quality levels greater than the quality level they requested, only raises the quality level of high buffer clients, and considers the overhead of DASH to produce healthy buffer levels without sacrificing utilization or quality levels. FairQ is configurable in terms of the low buffer threshold and whether or not quality smoothing is turned on or off. Turning quality smoothing on prevents clients from making quality jump larger than one quality level at a time which increases QoE but also increases the total number of quality switches. Turning it off lowers the number of quality switches but allows for large jumps in quality which lowers QoE.

In Chapter 6, we evaluated FairQ and FairQ-smooth (FairQ with quality smoothing turned on) under the same network and clients scenarios used in Chapter 4. We tested FairQ and FairQ-smooth using low buffer thresholds of 15, 20, and 25 seconds. We picked the best variation of FairQ under each client scenario and compared the results to the best algorithm for each metric in Chapter 4. We found the following: (1) FairQ produced the best fairness under mixed client scenarios with an average increase of 53% over the best algorithms from Chapter 4. (2) FairQ

produced the best utilization and quality levels for mixed client scenarios under Drop 1 and Big Drop with an average increase of 6% over the best algorithms from Chapter 4. FairQ also produced the second best utilization and quality levels under the Real Bandwidth Trace with a decrease of only 2% over the best algorithm from Chapter 4. (3) FairQ produced the lowest number of quality switches under Drop 1 and Big Drop network scenarios across all client scenarios with an average decrease of 34% over the best algorithms from Chapter 4.

7.2 Future Work

This research can be expanded upon in a variety of different ways starting with the experimental setup used for all the tests in this thesis. The experimental setup included the "Big Buck Bunny" video with a segment size of 2 seconds and each client was configured to have a buffer size of 30 seconds. These choices were made because they are very common in other research however, it may be beneficial to include more buffer levels, segment sizes, and videos. Different video types and segment sizes can affect encoding efficiency resulting in a greater range of segment sizes in bytes. By doing this, we could show the robustness of FairQ and how well it can adapt to different streaming scenarios.

Another area of expansion is the comparison algorithms used. We based our choice of comparison algorithm on previous research and by using the default algorithm in the video player we used. We included a rate-based, buffer-based, hybrid, resource fairness, and QoE fairness algorithm. It would beneficial to include more than just one algorithm in each category to get a more fair comparison. It would also be beneficial to compare our algorithm to not only academic algorithms presented in journals and conferences but also commercial algorithms that are used by large streaming services such as Netflix. By doing this, we could present this work as not just an academic exercise but a commercially viable approach to solving real world streaming problems that occur everyday.

FairQ clients utilize a simple algorithm that consists of the harmonic mean of the previous

20 throughput values as a bandwidth estimation. We could further experiment by using different bandwidth estimations on the client side while still using the FairQ server-side algorithm. This could include using not just the harmonic mean of throughput values but the arithmetic mean or the median. We could also include buffer levels into the bandwidth estimation or even insert FairQ on top of already proposed client side algorithms as this would only take a simple modification of these algorithms to send information to the server-side algorithm. By doing this, we could propose FairQ as not only a new adaptation algorithm but an adaptation scheme that can be inserted to already existing client-side adaptation algorithms to increase QoE fairness.

One issue with FairQ that was revealed in Chapter 6 is the high number of quality switches FairQ produced under the Real Bandwidth Trace. A contributing factor to this problem is that quality levels spiked directly after the initial phase and then returned to more appropriate levels within a few segment requests. This is a result of FairQ trying to use as much bandwidth as possible when some clients have just left the initial phase and others are still in the initial phase. When some clients are still in the initial phase, there is a lot of available bandwidth for clients just leaving the initial phase to use. To fix this issue, we could prevent clients from going above the fair quality level assigned to them by FairQ when there are still clients in the initial phase. This would reduce utilization slightly however, it would help address the large number of quality switches experienced by FairQ clients.

We could also expand the scenarios used to test FairQ and the comparison algorithms. We used a number of different manufactured dynamic scenarios to test for specific behaviors using 4 clients simultaneously. We also included a real bandwidth trace using 20 clients simultaneously. Under all these scenarios we configured all the clients to start and stop streaming at the same time. The use of real bandwidth trace helped showed the practicality of FairQ however, it would be beneficial to include more real bandwidth traces and a larger number of clients to further show the practicality of FairQ. It would also be beneficial to include scenarios where clients start and stop streaming at different times as this happens in the real world.

In addition to using more real bandwidth traces and a larger number of clients, we could also include more complex network topologies. We used two simple network toplogies, one with clients connecting directly to the server, and another with clients first connecting to a proxy that connects directly to the server used to implement server-side algorithms. Realistically, it would be unfeasible to run FairQ on a video server with thousands of clients streaming simultaneously as it would likely add overhead resulting in a decrease of QoE. To use FairQ in the real world it would need to be run at different bottlenecks within the network. We could include network toplogies that include two or more groups of clients that each connect to different proxies running their own instance of FairQ to better simulate these conditions. We could then scale up the number of proxies running concurrently to achieve much more realistic testing scenarios.

As briefly mentioned in Chapter 6, the CPU usage of FairQ on the proxy was on average 15.5% during a test using 20 client simultaneously while the average time the CPU spent idle was 95.9%. We are fairly satisfied with this performance however, this test only used 20 clients and more realistic scenarios would include hundreds and possibly thousands of concurrent clients. With more realistic client numbers the performance of FairQ may become an issue that affects algorithm performance. To address this, the FairQ implementation would need to be tested and any bottlenecks would need to be identified. FairQ is also implemented in python and it may be beneficial to implement FairQ in a lower level language such as `C` to improve performance. In either case, this is an important pieces of future work and would need to be addressed if FairQ were to be presented as a real world solution to any video streaming issues.

When determining quality levels, FairQ ensures that the sum of all client quality levels does not surpass the servers bandwidth. In this thesis, we simply set the server capacity prior to the test to either the bandwidth limit under static network conditions or the average over the length of the test for dynamic network conditions. We did this in order to focus on the adaptation algorithm and its impact on QoE and fairness however, this would need to be determined dynamically and not prior to the server being run in the real world. We could use an average or harmonic mean

of all currently connected clients throughput values to estimate server bandwidth or a tool that periodically tests network conditions such as the linux command line tool `iftop`. This is one of the main implementation issues of FairQ and would need to be thoroughly investigated in any future work poorly estimating server bandwidth would could lead to very poor algorithm performance across all clients.

Bibliography

[1] Adobe. Adobe http dynamic streaming. Technical report, Adobe, 2019.

[2] 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, New York, NY, USA, 2012. ACM.

[3] Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, and Erik Elmroth. Analysis and characterization of a video-on-demand service workload. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 189–200, New York, NY, USA, 2015. ACM.

[4] Ramon Aparicio-Pardo, Karine Pires, Alberto Blanc, and Gwendal Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 49–60, New York, NY, USA, 2015. ACM.

[5] Jiasi Chen, Rajesh Mahindra, Mohammad Amir Khojastepour, Sampath Rangarajan, and Mung Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, pages 389–400, New York, NY, USA, 2013. ACM.

[6] Federico Chiariotti, Stefano D'Aronco, Laura Toni, and Pascal Frossard. Online learning adaptation strategy for dash clients. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 8:1–8:12, New York, NY, USA, 2016. ACM.

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

[8] Cisco. Cisco visual networking index: Forecast and trends, 20172022 white paper. Technical report, Cisco, 2017.

[9] M. Claeys, S. Latre, J. Famaey, and F. De Turck. Design and evaluation of a self-learning http adaptive video streaming client. *IEEE Communications Letters*, 18(4):716–719, April 2014.

[10] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. Design and experimental evaluation of network-assisted strategies for http adaptive streaming. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 3:1–3:12, New York, NY, USA, 2016. ACM.

[11] Nicola Cranley, Philip Perry, and Liam Murphy. User perception of adapting video quality. *Int. J. Hum.-Comput. Stud.*, 64(8):637–647, August 2006.

[12] FFmpeg Developers. *ffmpeg - A complete, cross-platform solution to record, convert and stream audio and video*. http://ffmpeg.org/, December 2016.

[13] R. Dubin, O. Hadar, and A. Dvir. The effect of client buffer and mbr consideration on dash adaptation logic. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2178–2183, April 2013.

[14] 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.

[15] Fanqing Gao, Guangjun Wen, Zhengyong Feng, and Zixuan Zou. Improvement on frame layer rate control scheme for h.264/avc. In *2009 IEEE International Conference on Network Infrastructure and Digital Content*, pages 666–669, Nov 2009.

[16] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In

Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking, FhMN '13, pages 15–20, New York, NY, USA, 2013. ACM.

[17] Tobias Hoíßfeld, Michael Seufert, Christian Sieber, Thomas Zinner, and Phuoc Tran-Gia. Identifying qoe optimal adaptation of http adaptive streaming based on subjective studies. *Comput. Netw.*, 81(C):320–332, April 2015.

[18] 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.

[19] Zixia Huang, Chao Mei, Li Li, and T. Woo. Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy. In *INFOCOM, 2011 Proceedings IEEE*, pages 201–205, April 2011.

[20] ISO/IEC. Information technology - Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats. Technical Report ISO 23009-1, ISO, Geneva, Switzerland, 2014.

[21] 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):326–340, Feb 2014.

[22] 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.

[23] Keunsoo Kim, Benjamin Y. Cho, and Won Woo Ro. Server side, play buffer based quality control for adaptive media streaming. *Multimedia Tools Appl.*, 75(10):5397–5415, May 2016.

[24] Dilip Kumar Krishnappa, Michael Zink, and Ramesh K. Sitaraman. Optimizing the video transcoding workflow in content delivery networks. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 37–48, New York, NY, USA, 2015. ACM.

[25] Alexey N. Kuznetsov. *tc - show / manipulate traffic control settings*, December 2016.

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

[27] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. Gpac: Open source multimedia framework. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 1009–1012, New York, NY, USA, 2007. ACM.

[28] 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, NY, USA, 2012. ACM.

[29] Yunlong Li, Yue Wang, Shanshe Wang, and Siwei Ma. An adaptative bitrate algorithm for dash. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[30] 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.

[31] Q. Lin, Y. Liu, Y. Shen, H. Shen, L. Sang, and D. Yang. Bandwidth estimation of rate adaption algorithm in dash. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 243–247, Dec 2014.

[32] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate adaptation for adaptive http streaming. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*,

MMSys '11, pages 169–174, New York, NY, USA, 2011. ACM.

[33] He Ma, Beomjoo Seo, and Roger Zimmermann. Dynamic scheduling on video transcoding for mpeg dash in the cloud environment. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, pages 283–294, New York, NY, USA, 2014. ACM.

[34] V. Menkovski and A. Liotta. Intelligent control for adaptive video streaming. In *2013 IEEE International Conference on Consumer Electronics (ICCE)*, pages 127–128, Jan 2013.

[35] Microsoft. Microsoft smooth streaming. Technical report, Microsoft, 2019.

[36] Ricky K. P. Mok, Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. Qdash: A qoe-aware dash system. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 11–22, New York, NY, USA, 2012. ACM.

[37] Ricky K.P. Mok, Edmond W.W. Chan, Xiapu Luo, and Rocky K.C. Chang. Inferring the qoe of http video streaming from user-viewing activities. In *Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack*, W-MUST '11, pages 31–36, New York, NY, USA, 2011. ACM.

[38] Emanuele Oriani. *qpsnr - quick psnr*. http://qpsnr.youlink.org/, February 2015.

[39] H. Parmar and M. Thornburgh. Adobes real time messaging protocol. Technical report, Adobe, United States, 2012.

[40] S. Petrangeli, M. Claeys, S. Latr, J. Famaey, and F. De Turck. A multi-agent q-learning-based framework for achieving fairness in http adaptive streaming. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9, May 2014.

[41] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. Qoe-driven rate adaptation heuristic for fair adaptive video streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, 12(2):28:1–28:24, October 2015.

[42] Slawomir Przylucki, Dariusz Czerwinski, and Artur Sierszen. Qoe-oriented fairness control for dash systems based on the hierarchical structure of svc streams. In Piotr Gaj, Andrzej Kwiecień, and Piotr Stera, editors, *Computer Networks*, pages 180–191, Cham, 2016. Springer International Publishing.

[43] Wei Pu, Zixuan Zou, and Chang Wen Chen. Video adaptation proxy for wireless dynamic adaptive streaming over http. In *2012 19th International Packet Video Workshop (PV)*, pages 65–70, May 2012.

[44] L. Qian, Z. Li, P. Zhou, and J. Chen. An improved matrix encoding steganography algorithm based on h.264 video. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 256–260, June 2016.

[45] 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, pages 460–465, New York, NY, USA, 2018. ACM.

[46] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla. Sdn based qoe optimization for http-based adaptive video streaming. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 120–123, Dec 2015.

[47] Demóstenes Z. Rodríguez, Zhou Wang, Renata L. Rosa, and Graça Bressan. The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over http. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):216, Dec 2014.

[48] Sandvine. 2018 global internet phenomena report. Technical report, Sandvine, 2018.

[49] 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.

[50] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. Technical report, RFC Editor, United States, 2003.

[51] Thiago Silva and Dorgival Guedes. Live streaming of user generated videos: Workload characterization and content delivery architectures. *Computer Networks*, 55:4055–4068, 12 2011.

[52] 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.

[53] J. Summers, T. Brecht, D. Eager, and A. Gutarin. Characterizing the workload of a netflix streaming video server. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–12, Sep. 2016.

[54] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados. A control-based algorithm for rate adaption in mpeg-dash. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, pages 438–442, July 2014.

[55] S. Wang, A. Rehman, Z. Wang, S. Ma, and W. Gao. Rate-ssim optimization for video coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 833–836, May 2011.

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

[57] W. Wu and X. Zhang. Code performance improvement scheme for x264 based on ssim. In *2012 3rd IEEE International Conference on Network Infrastructure and Digital Content*, pages 396–400, Sep. 2012.

[58] 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.

[59] A. H. Zahran, D. Raca, and C. J. Sreenan. Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks. *IEEE Transactions on Mobile Computing*, 17(12):2716–2728, Dec 2018.

[60] C. Zhou, C. W. Lin, and Z. Guo. mdash: A markov decision-based rate adaptation approach for dynamic http streaming. *IEEE Transactions on Multimedia*, 18(4):738–751, April 2016.

[61] Y. Zhou, Y. Duan, J. Sun, and Z. Guo. Towards simple and smooth rate adaption for vbr video in dash. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 9–12, Dec 2014.

[62] Zhou Wang, Ligang Lu, and A. C. Bovik. Video quality assessment using structural distortion measurement. In *Proceedings. International Conference on Image Processing*, volume 3, pages III–III, Sep. 2002.

Appendix A

Overall Results from Chapter 4 and 6

This appendix contains the results in their entirety from the tests performed for Chapter 4 and 6. Each table contains the results for all variations of FairQ tested and all the comparison algorithms used in Chapter 4. The best result under each metric will be listed in bold with multiple values being listed in bold in the case of ties.

A.1 Bandwidth Drops

A.1.1 Homogeneous Client Scenario

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000853	78.3	4.3	27.3	**0.6**	2.056
Buffer	0.001680	86.8	24.0	25.6	16.6	2.278
Rate	0.001488	**91.9**	9.9	26.4	41.8	**2.411**
FESTIVE	**0.000438**	36.7	15.5	**29.0**	1.7	0.963
Hybrid	0.002561	53.3	15.8	23.8	**0.6**	1.400
FairQ-smooth 15s	0.001616	79.4	4.5	28.2	13.4	2.083
FairQ 15s	0.001370	80.3	3.0	27.9	13.4	2.108
FairQ-smooth 20s	0.002102	79.3	4.3	28.2	13.5	2.081
FairQ 20s	0.001543	80.4	**2.7**	27.9	13.2	2.109
FairQ-smooth 25s	0.001579	79.3	4.7	28.2	13.6	2.082
FairQ 25s	0.001970	80.1	2.9	28.0	13.5	2.103

Table A.1: Results from the Drop 1 network scenario under a homogeneous client scenario.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000713	78.5	5.0	27.4	**0.6**	2.060
Buffer	0.001029	86.0	23.2	26.0	17.3	2.257
Rate	0.001438	**91.7**	9.8	26.2	41.6	**2.408**
FESTIVE	**0.000460**	34.9	15.0	**29.1**	1.6	0.916
Hybrid	0.002138	55.5	17.6	24.8	**0.6**	1.456
FairQ-smooth 15s	0.001414	79.5	5.3	28.3	13.4	2.086
FairQ 15s	0.001920	80.4	4.0	27.8	13.6	2.110
FairQ-smooth 20s	0.002188	79.3	4.0	28.3	13.3	2.082
FairQ 20s	0.001929	80.2	3.2	27.9	13.6	2.105
FairQ-smooth 25s	0.000731	79.5	4.7	28.2	13.4	2.086
FairQ 25s	0.001332	80.2	**2.8**	28.1	13.8	2.106

Table A.2: Results from the Drop 2 network scenario under a homogeneous client scenario.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.001842	79.2	**3.2**	27.6	**0.6**	1.882
Buffer	0.001878	90.0	25.2	25.0	16.7	2.139
Rate	0.001365	**94.9**	11.9	26.0	44.0	**2.254**
FESTIVE	0.000388	37.7	15.5	**28.9**	1.2	0.895
Hybrid	0.001893	57.2	16.8	24.3	**0.6**	1.359
FairQ-smooth 15s	0.000185	81.6	5.7	27.9	13.5	1.939
FairQ 15s	0.000515	82.1	3.7	27.8	13.5	1.949
FairQ-smooth 20s	**0.000138**	81.5	5.5	28.0	13.5	1.936
FairQ 20s	0.000522	82.1	3.8	27.8	13.6	1.949
FairQ-smooth 25s	0.000143	81.7	5.7	27.9	13.2	1.941
FairQ 25s	0.000413	81.8	6.3	27.8	13.4	1.943

Table A.3: Results from the Drop 3 network scenario under a homogeneous client scenario.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000935	80.5	4.6	27.5	**0.6**	1.912
Buffer	0.000667	88.3	26.1	25.9	17.9	2.096
Rate	0.000598	**93.7**	12.2	25.9	41.8	**2.225**
FESTIVE	0.000339	37.4	15.7	**28.9**	1.6	0.889
Hybrid	0.001952	56.5	15.8	24.0	**0.6**	1.341
FairQ-smooth 15s	0.000304	82.0	5.7	28.2	13.9	1.947
FairQ 15s	0.000623	82.9	3.8	27.8	13.9	1.968
FairQ-smooth 20s	**0.000215**	81.7	8.0	28.0	14.4	1.940
FairQ 20s	0.000657	82.8	4.5	27.7	13.9	1.966
FairQ-smooth 25s	0.000614	82.1	6.5	28.1	13.8	1.950
FairQ 25s	0.000610	82.8	**3.4**	27.8	13.6	1.967

Table A.4: Results from the Drop 4 network scenario under a homogeneous client scenario.

A.1.2 Mixed Client Scenario

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.005653	74.8	1.2	28.6	**0.6**	1.963
Buffer	0.009671	68.1	10.7	28.6	10.1	1.789
Rate	0.007309	73.6	5.7	28.4	24.5	1.932
FESTIVE	0.032230	25.2	11.0	**29.4**	1.2	0.661
Hybrid	0.022612	40.7	13.0	24.6	**0.6**	1.068
FairQ-smooth 15s	0.002470	79.7	4.5	28.7	9.7	2.092
FairQ 15s	0.002194	80.7	**1.0**	28.6	9.6	2.118
FairQ-smooth 20s	0.002562	80.0	4.5	28.7	9.4	2.099
FairQ 20s	**0.002136**	80.7	**1.0**	28.5	9.8	2.118
FairQ-smooth 25s	0.002446	79.9	4.5	28.7	9.5	2.098
FairQ 25s	0.002235	**80.8**	**1.0**	28.4	9.4	**2.120**

Table A.5: Results from the Drop 1 network scenario under a mixed client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.005470	74.9	**1.0**	28.6	**0.6**	1.967
Buffer	0.009789	68.2	11.5	28.7	10.1	1.791
Rate	0.008027	71.8	5.8	28.6	23.2	1.886
FESTIVE	0.031790	25.8	11.3	**29.4**	1.9	0.678
Hybrid	0.022824	40.4	13.0	24.7	**0.6**	1.060
FairQ-smooth 15s	0.002425	79.8	4.8	28.6	9.8	2.095
FairQ 15s	0.002251	**80.8**	**1.0**	28.4	9.6	**2.121**
FairQ-smooth 20s	0.002557	79.9	4.5	28.6	9.5	2.099
FairQ 20s	**0.002178**	80.7	**1.0**	28.5	9.9	2.120
FairQ-smooth 25s	0.002459	79.8	4.5	28.6	9.8	2.095
FairQ 25s	0.002251	**80.8**	**1.0**	28.4	9.9	**2.121**

Table A.6: Results from the Drop 2 network scenario under a mixed client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.006937	80.4	4.3	28.4	**0.6**	1.909
Buffer	0.011124	70.8	12.8	28.2	10.2	1.682
Rate	0.010535	72.8	7.2	28.3	25.3	1.728
FESTIVE	0.032127	27.6	11.2	**29.3**	1.5	0.655
Hybrid	0.023374	43.8	12.8	24.6	**0.6**	1.041
FairQ-smooth 15s	0.003217	81.9	6.6	28.5	7.0	1.946
FairQ 15s	0.001415	82.8	3.2	28.3	7.1	1.967
FairQ-smooth 20s	**0.001252**	81.4	7.3	28.4	7.1	1.934
FairQ 20s	0.003164	**82.9**	**2.6**	28.2	7.1	**1.969**
FairQ-smooth 25s	0.003208	81.6	6.6	28.5	7.0	1.939
FairQ 25s	0.001526	82.8	3.4	28.2	7.3	1.967

Table A.7: Results from the Drop 3 network scenario under a mixed client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.005766	82.1	3.2	28.3	0.7	1.950
Buffer	0.011572	69.7	13.2	28.6	10.5	1.656
Rate	0.010903	71.6	7.4	28.3	23.6	1.701
FESTIVE	0.032294	27.4	11.0	**29.3**	1.7	0.650
Hybrid	0.024008	42.1	12.8	24.6	**0.6**	1.000
FairQ-smooth 15s	**0.001052**	82.3	6.4	28.6	7.1	1.955
FairQ 15s	0.003635	83.5	**2.2**	28.3	6.9	1.983
FairQ-smooth 20s	0.003246	82.2	7.1	28.6	7.0	1.953
FairQ 20s	0.003565	**83.7**	2.3	28.2	7.0	**1.988**
FairQ-smooth 25s	0.003687	82.5	5.8	28.6	6.8	1.960
FairQ 25s	0.001476	83.6	2.7	28.3	7.1	1.985

Table A.8: Results from the Drop 4 network scenario under a mixed client scenario.

A.1.3 Homogeneous Within Mixed Client Scenario

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00040102	68.2	1.3	28.3	**0.6**	3.583
Buffer	0.00020484	60.6	19.3	27.7	15.8	3.183
Rate	0.00062640	66.1	10.3	27.4	43.8	3.469
FESTIVE	0.00007297	18.8	15.0	**29.2**	1.2	0.986
Hybrid	0.00101045	34.4	19.0	24.5	0.7	1.808
FairQ-smooth 15s	0.00010583	72.9	5.0	28.3	16.3	3.826
FairQ 15s	0.00000860	73.8	**1.0**	28.1	16.3	3.873
FairQ-smooth 20s	0.00000989	73.1	5.0	28.4	15.7	3.840
FairQ 20s	0.00000860	73.8	**1.0**	28.1	16.4	3.873
FairQ-smooth 25s	0.00005318	73.1	5.0	28.4	15.8	3.837
FairQ 25s	**0.00000000**	**73.9**	**1.0**	27.9	16.1	**3.878**

Table A.9: Results from the Drop 1 network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.00000000**	6.5	**1.0**	29.0	0.6	0.343
Buffer	0.00000084	**7.5**	2.0	**29.6**	4.3	**0.395**
Rate	0.00000075	**7.5**	**1.0**	29.4	5.2	**0.395**
FESTIVE	0.00005232	6.4	7.0	29.5	1.2	0.336
Hybrid	0.00019153	6.2	7.0	24.8	**0.5**	0.328
FairQ-smooth 15s	0.00019804	6.8	4.0	29.0	3.1	0.358
FairQ 15s	0.00008314	6.9	**1.0**	29.0	2.9	0.363
FairQ-smooth 20s	0.00021503	6.8	4.0	29.0	3.0	0.358
FairQ 20s	**0.00000000**	6.9	**1.0**	29.0	3.2	0.363
FairQ-smooth 25s	**0.00000000**	6.8	4.0	29.0	3.1	0.359
FairQ 25s	0.00011425	6.9	**1.0**	28.9	2.8	0.363

Table A.10: Results from the Drop 1 network scenario under a mixed client scenario showing only the 360p clients.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.00000000**	68.4	**1.0**	28.1	**0.7**	3.590
Buffer	0.00157863	60.7	21.0	27.8	15.8	3.187
Rate	0.00063864	64.3	10.5	27.7	41.0	3.376
FESTIVE	**0.00000000**	19.4	15.7	**29.3**	1.8	1.020
Hybrid	0.00009676	34.1	19.0	24.6	**0.7**	1.792
FairQ-smooth 15s	0.00005740	73.0	5.7	28.3	16.6	3.831
FairQ 15s	**0.00000000**	**73.9**	**1.0**	27.8	16.3	**3.879**
FairQ-smooth 20s	**0.00000000**	73.1	5.0	28.3	16.0	3.839
FairQ 20s	0.00004405	73.8	**1.0**	28.0	16.5	3.876
FairQ-smooth 25s	**0.00000000**	73.0	5.0	28.2	16.5	3.832
FairQ 25s	0.00001737	**73.9**	**1.0**	27.8	16.4	**3.879**

Table A.11: Results from the Drop 2 network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00009270	6.5	**1.0**	29.1	**0.5**	0.343
Buffer	0.00000210	**7.5**	2.0	**29.5**	4.4	**0.395**
Rate	0.00000299	**7.5**	**1.0**	**29.5**	5.4	**0.395**
FESTIVE	0.00007285	6.4	7.0	**29.5**	1.9	0.337
Hybrid	0.00033443	6.3	7.0	24.8	**0.5**	0.328
FairQ-smooth 15s	0.00013299	6.8	4.0	29.0	3.0	0.359
FairQ 15s	0.00009902	6.9	**1.0**	28.9	2.9	0.363
FairQ-smooth 20s	0.00021503	6.8	4.0	29.0	3.1	0.358
FairQ 20s	**0.00000000**	6.9	**1.0**	28.9	3.2	0.363
FairQ-smooth 25s	0.00009902	6.8	4.0	28.9	3.0	0.358
FairQ 25s	0.00009902	6.9	**1.0**	28.9	3.4	0.363

Table A.12: Results from the Drop 2 network scenario under a mixed client scenario showing only the 360p clients.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.00001851**	73.2	7.7	28.1	**0.7**	3.476
Buffer	0.00069288	62.5	23.5	26.8	15.5	2.968
Rate	0.00034528	64.4	13.3	27.1	44.0	3.061
FESTIVE	0.00015527	20.6	15.3	**29.2**	1.5	0.976
Hybrid	0.00159792	36.9	18.5	24.5	**0.7**	1.755
FairQ-smooth 15s	0.00450240	74.6	8.2	28.1	12.0	3.543
FairQ 15s	0.00085456	75.3	5.3	27.8	12.4	3.578
FairQ-smooth 20s	0.00147296	74.1	9.7	28.0	12.2	3.518
FairQ 20s	0.00408836	**75.4**	**4.2**	27.6	12.5	**3.582**
FairQ-smooth 25s	0.00449420	74.3	8.2	28.1	12.1	3.529
FairQ 25s	0.00130065	75.3	5.8	27.7	12.4	3.578

Table A.13: Results from the Drop 3 network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00030998	7.2	**1.0**	28.8	**0.5**	0.342
Buffer	0.00000042	**8.3**	2.0	**29.6**	4.8	**0.395**
Rate	0.00000114	**8.3**	**1.0**	29.5	6.7	**0.395**
FESTIVE	0.00010715	7.0	7.0	29.4	1.5	0.334
Hybrid	0.00089434	6.9	7.0	24.7	**0.5**	0.328
FairQ-smooth 15s	0.00009092	7.4	5.0	28.8	1.9	0.350
FairQ 15s	**0.00000000**	7.5	**1.0**	28.8	1.8	0.356
FairQ-smooth 20s	0.00001985	7.3	5.0	28.9	2.0	0.349
FairQ 20s	0.00012871	7.5	**1.0**	28.8	1.8	0.356
FairQ-smooth 25s	0.00011078	7.3	5.0	28.9	1.9	0.349
FairQ 25s	0.00003482	7.5	**1.0**	28.8	2.2	0.355

Table A.14: Results from the Drop 3 network scenario under a mixed client scenario showing only the 360p clients.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00047032	74.9	5.5	27.8	**0.6**	3.559
Buffer	0.00009911	61.4	24.3	27.6	16.2	2.918
Rate	0.00057574	63.3	13.8	27.0	40.9	3.006
FESTIVE	**0.00007903**	20.3	15.0	**29.2**	1.7	0.966
Hybrid	0.00132412	35.2	18.5	24.3	0.7	1.671
FairQ-smooth 15s	0.00146454	75.0	7.8	28.3	12.1	3.561
FairQ 15s	0.00506771	76.0	**3.5**	27.6	12.0	3.611
FairQ-smooth 20s	0.00459007	74.9	9.2	28.3	12.2	3.558
FairQ 20s	0.00492703	**76.2**	3.7	27.6	12.2	**3.620**
FairQ-smooth 25s	0.00521086	75.2	6.5	28.3	11.8	3.570
FairQ 25s	0.00176771	76.1	4.3	27.7	12.4	3.614

Table A.15: Results from the Drop 4 network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00057272	7.2	**1.0**	28.9	0.7	0.342
Buffer	**0.00000000**	**8.3**	2.0	**29.5**	4.8	**0.395**
Rate	0.00000001	**8.3**	**1.0**	**29.5**	6.3	**0.395**
FESTIVE	0.00005228	7.0	7.0	29.4	1.7	0.334
Hybrid	0.00005699	6.9	7.0	24.8	**0.5**	0.329
FairQ-smooth 15s	0.00013168	7.4	5.0	28.9	2.1	0.349
FairQ 15s	**0.00000000**	7.5	**1.0**	28.9	1.7	0.354
FairQ-smooth 20s	**0.00000000**	7.3	5.0	28.9	1.8	0.348
FairQ 20s	**0.00000000**	7.5	**1.0**	28.9	1.8	0.356
FairQ-smooth 25s	0.00020170	7.3	5.0	28.9	1.8	0.349
FairQ 25s	**0.00000000**	7.5	**1.0**	28.9	1.8	0.356

Table A.16: Results from the Drop 4 network scenario under a mixed client scenario showing only the 360p clients.

A.2 Bandwidth Drops and Raises

A.2.1 Homogeneous Client Scenarios

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.002014	78.8	3.3	27.7	**0.7**	1.904
Buffer	0.000837	86.8	22.9	26.2	18.4	2.097
Rate	0.000889	**93.6**	10.1	25.7	40.0	**2.263**
FESTIVE	0.000519	37.4	15.5	**28.9**	1.4	0.903
Hybrid	0.001496	56.1	16.2	24.2	**0.7**	1.356
FairQ-smooth 15s	**0.000135**	80.4	5.2	28.3	15.5	1.943
FairQ 15s	0.000263	81.0	**1.7**	28.2	15.7	1.958
FairQ-smooth 20s	0.000666	80.9	5.7	28.3	16.1	1.955
FairQ 20s	0.000407	81.2	2.1	28.2	15.9	1.961
FairQ-smooth 25s	0.000280	80.3	5.2	28.3	15.6	1.941
FairQ 25s	0.000182	81.0	3.2	28.2	15.4	1.958

Table A.17: Results from the Big Drop network scenario under a homogeneous client scenario.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.001341	75.6	5.5	27.1	**0.7**	1.827
Buffer	0.001692	86.2	25.3	25.7	18.4	2.084
Rate	0.000797	**91.3**	10.2	26.1	42.5	**2.208**
FESTIVE	**0.000127**	37.8	15.3	**28.9**	1.0	0.915
Hybrid	0.001851	57.8	16.2	24.0	**0.7**	1.398
FairQ-smooth 15s	0.001041	79.2	7.2	27.8	15.6	1.915
FairQ 15s	0.000474	80.2	**5.1**	27.3	16.3	1.938
FairQ-smooth 20s	0.000348	78.8	7.8	27.8	15.4	1.905
FairQ 20s	0.000949	79.4	5.8	27.4	15.8	1.919
FairQ-smooth 25s	0.000613	78.7	8.5	27.8	15.8	1.901
FairQ 25s	0.000501	79.4	5.5	27.7	15.8	1.919

Table A.18: Results from the Big Raise network scenario under a homogeneous client scenario.

Homogeneous	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000693	77.4	4.5	26.8	**0.7**	1.935
Buffer	0.002083	85.7	23.4	25.8	18.2	2.142
Rate	0.001156	**89.1**	9.4	26.2	38.7	**2.227**
FESTIVE	**0.000067**	36.6	15.0	**29.1**	2.0	0.915
Hybrid	0.000536	56.8	15.9	23.8	**0.7**	1.420
FairQ-smooth 15s	0.001510	79.4	6.6	27.7	15.5	1.984
FairQ 15s	0.000828	80.2	5.1	27.3	15.8	2.004
FairQ-smooth 20s	0.001739	79.8	6.8	27.7	15.7	1.994
FairQ 20s	0.001791	80.5	4.6	27.5	15.8	2.013
FairQ-smooth 25s	0.001746	79.5	6.9	27.5	15.5	1.988
FairQ 25s	0.001546	80.1	**4.4**	27.6	16.0	2.003

Table A.19: Results from the Drop and Raise network scenario under a homogeneous client scenario.

A.2.2 Mixed Client Scenarios

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.006973	79.6	2.5	28.5	0.8	1.923
Buffer	0.011472	69.5	12.8	28.4	11.3	1.680
Rate	0.009952	72.1	7.3	28.2	23.3	1.743
FESTIVE	0.032365	27.0	11.0	**29.3**	2.0	0.653
Hybrid	0.023902	40.9	13.0	24.4	**0.7**	0.988
FairQ-smooth 15s	0.001639	80.6	5.7	28.6	8.2	1.947
FairQ 15s	0.004206	**82.2**	**1.1**	28.5	8.4	**1.987**
FairQ-smooth 20s	0.001282	81.0	5.6	28.6	8.1	1.957
FairQ 20s	0.001405	**82.2**	1.7	28.5	8.3	1.986
FairQ-smooth 25s	**0.001262**	80.9	5.7	28.6	8.1	1.955
FairQ 25s	0.003639	82.1	1.5	28.5	8.5	1.984

Table A.20: Results from the Big Drop network scenario under a mixed client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.007062	79.2	3.5	28.2	**0.7**	1.913
Buffer	0.012333	66.9	14.1	28.1	11.1	1.616
Rate	0.010890	70.2	7.2	28.5	24.0	1.698
FESTIVE	0.031849	28.0	11.2	**29.4**	1.8	0.678
Hybrid	0.025346	38.9	12.7	24.6	**0.7**	0.939
FairQ-smooth 15s	**0.001171**	79.2	7.9	28.5	7.9	1.914
FairQ 15s	0.002867	80.7	**3.3**	28.2	8.3	1.951
FairQ-smooth 20s	0.001622	79.7	8.2	28.6	7.9	1.927
FairQ 20s	0.001622	**81.0**	3.7	28.2	8.3	**1.958**
FairQ-smooth 25s	0.001479	79.4	8.6	28.5	8.1	1.919
FairQ 25s	0.002057	80.5	4.2	28.1	8.3	1.946

Table A.21: Results from the Big Raise network scenario under a mixed client scenario.

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.006938	76.5	2.8	28.4	**0.7**	1.913
Buffer	0.011315	67.3	12.5	27.9	11.1	1.682
Rate	0.010142	69.2	6.6	28.5	22.0	1.731
FESTIVE	0.031965	26.5	11.2	**29.4**	1.4	0.663
Hybrid	0.023330	40.8	13.0	24.8	**0.7**	1.021
FairQ-smooth 15s	0.003240	78.0	6.8	28.6	8.0	1.949
FairQ 15s	0.003522	**78.9**	**2.7**	28.5	8.2	**1.972**
FairQ-smooth 20s	**0.001127**	77.4	7.1	28.5	8.2	1.935
FairQ 20s	0.001687	78.1	2.8	28.5	8.6	1.952
FairQ-smooth 25s	0.001183	78.0	6.8	28.6	8.1	1.949
FairQ 25s	0.003097	**78.9**	3.2	28.4	8.2	**1.972**

Table A.22: Results from the Drop and Raise network scenario under a mixed client scenario.

A.2.3 Homogeneous Within Mixed Client Scenarios

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.00040102**	72.5	4.0	28.1	1.1	3.504
Buffer	0.00149149	61.3	23.5	27.1	17.8	2.964
Rate	0.00135650	63.9	13.7	26.9	40.4	3.090
FESTIVE	0.00042059	20.1	15.0	**29.2**	2.0	0.971
Hybrid	0.00127333	34.1	19.0	24.2	**0.9**	1.649
FairQ-smooth 15s	0.00228569	73.3	6.3	28.2	14.3	3.544
FairQ 15s	0.00589855	**74.9**	**1.2**	28.1	14.3	**3.619**
FairQ-smooth 20s	0.00181169	73.8	6.2	28.2	13.9	3.566
FairQ 20s	0.00175828	74.8	2.3	28.1	14.2	3.616
FairQ-smooth 25s	0.00177604	73.6	6.3	28.3	14.2	3.559
FairQ 25s	0.00509816	74.8	2.0	28.1	14.4	3.614

Table A.23: Results from the Big Drop network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	**0.00000000**	7.1	**1.0**	28.9	**0.6**	0.342
Buffer	0.00000084	**8.2**	2.0	**29.7**	4.9	**0.395**
Rate	0.00000297	**8.2**	**1.0**	29.5	6.2	**0.395**
FESTIVE	0.00011005	6.9	7.0	29.5	2.1	0.335
Hybrid	0.00014675	6.8	7.0	24.6	**0.6**	0.327
FairQ-smooth 15s	0.00024246	7.2	5.0	29.0	2.1	0.349
FairQ 15s	0.00009390	7.4	**1.0**	29.0	2.4	0.356
FairQ-smooth 20s	0.00004076	7.2	5.0	28.9	2.2	0.349
FairQ 20s	0.00009495	7.4	**1.0**	28.9	2.5	0.356
FairQ-smooth 25s	0.00001985	7.2	5.0	29.0	2.1	0.350
FairQ 25s	0.00005275	7.3	**1.0**	28.9	2.6	0.354

Table A.24: Results from the Big Drop network scenario under a mixed client scenario showing only the 360p clients.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00030601	72.1	6.0	27.4	0.8	3.484
Buffer	0.00003619	58.7	26.2	26.7	16.6	2.837
Rate	0.00021158	62.1	13.5	27.6	41.9	3.000
FESTIVE	**0.00006911**	21.1	15.5	**29.3**	1.9	1.018
Hybrid	0.00314608	32.1	18.3	24.4	**0.7**	1.552
FairQ-smooth 15s	0.00066533	72.0	10.8	28.1	13.8	3.481
FairQ 15s	0.00323543	73.4	**5.7**	27.5	14.5	3.547
FairQ-smooth 20s	0.00186348	72.5	11.5	28.2	14.0	3.504
FairQ 20s	0.00136318	**73.7**	6.3	27.5	14.6	**3.561**
FairQ-smooth 25s	0.00132541	72.2	12.2	28.2	14.1	3.488
FairQ 25s	0.00122702	73.2	7.5	27.3	14.2	3.536

Table A.25: Results from the Big Raise network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00030998	7.1	**1.0**	28.9	**0.6**	0.343
Buffer	0.00000083	**8.2**	2.0	**29.6**	5.5	**0.395**
Rate	0.00000185	**8.2**	**1.0**	29.5	6.1	**0.395**
FESTIVE	0.00003907	7.0	7.0	29.5	1.8	0.337
Hybrid	0.00024719	6.8	7.0	24.8	**0.6**	0.326
FairQ-smooth 15s	0.00022155	7.2	5.0	28.8	2.0	0.348
FairQ 15s	0.00001688	7.4	**1.0**	28.9	2.1	0.356
FairQ-smooth 20s	**0.00000000**	7.2	5.0	28.9	1.8	0.349
FairQ 20s	**0.00000000**	7.4	**1.0**	28.9	2.1	0.355
FairQ-smooth 25s	0.00009092	7.2	5.0	28.8	2.2	0.350
FairQ 25s	0.00023949	7.3	**1.0**	28.9	2.3	0.355

Table A.26: Results from the Big Raise network scenario under a mixed client scenario showing only the 360p clients.

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00031736	69.7	4.7	27.8	**0.7**	3.484
Buffer	0.00035428	59.4	23.0	26.1	16.8	2.968
Rate	0.00093724	61.3	12.2	27.4	37.8	3.067
FESTIVE	**0.00022086**	19.8	15.3	**29.3**	1.3	0.991
Hybrid	0.00152972	34.3	19.0	24.7	0.8	1.714
FairQ-smooth 15s	0.00453637	71.0	8.5	28.3	13.8	3.548
FairQ 15s	0.00473865	**71.8**	**4.3**	28.1	14.4	3.588
FairQ-smooth 20s	0.00127865	70.4	9.2	28.1	13.8	3.521
FairQ 20s	0.00138441	71.0	4.7	28.0	15.3	3.548
FairQ-smooth 25s	0.00159850	71.0	8.5	28.3	13.9	3.550
FairQ 25s	0.00412143	**71.8**	5.5	28.0	14.1	**3.589**

Table A.27: Results from the Drop and Raise network scenario under a mixed client scenario showing only the 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.00030998	6.9	**1.0**	28.9	**0.6**	0.343
Buffer	0.00000212	**7.9**	2.0	**29.7**	5.4	**0.395**
Rate	**0.00000151**	**7.9**	**1.0**	29.5	6.2	**0.395**
FESTIVE	0.00000664	6.7	7.0	29.4	1.4	0.335
Hybrid	0.00043208	6.5	7.0	24.8	**0.6**	0.327
FairQ-smooth 15s	0.00001269	7.0	5.2	29.0	2.2	0.350
FairQ 15s	0.00011183	7.1	**1.0**	28.9	2.0	0.356
FairQ-smooth 20s	0.00013063	7.0	5.0	28.9	2.6	0.349
FairQ 20s	0.00011078	7.1	**1.0**	28.9	1.9	0.356
FairQ-smooth 25s	0.00025515	7.0	5.2	29.0	2.3	0.349
FairQ 25s	0.00020467	7.1	**1.0**	28.9	2.2	0.355

Table A.28: Results from the Drop and Raise network scenario under a mixed client scenario showing only the 360p clients.

A.3 Real Bandwidth Trace

A.3.1 Mixed Client Scenarios

Mixed	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.014651	74.1	**4.5**	25.8	1.3	0.562
Buffer	0.038831	77.1	27.6	24.9	19.2	0.585
Rate	0.032321	**82.8**	11.6	22.6	26.3	**0.628**
FESTIVE	0.039453	46.6	14.4	**26.9**	**1.2**	0.353
Hybrid	0.039374	68.1	12.1	22.7	1.3	0.516
FairQ-smooth 15s	0.006946	80.0	16.8	26.1	6.0	0.607
FairQ 15s	**0.005258**	81.4	12.4	26.1	7.5	0.617
FairQ-smooth 20s	0.005925	79.2	17.9	26.0	5.9	0.601
FairQ 20s	0.006188	81.5	13.2	26.0	7.5	0.618
FairQ-smooth 25s	0.008634	79.1	20.0	26.1	5.9	0.600
FairQ 25s	0.007739	80.8	13.2	26.2	7.6	0.613

Table A.29: Results from the Real Bandwidth Trace network scenario under a mixed client scenario.

A.3.2 Homogeneous Within Mixed Client Scenario

1080p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.001378	59.1	**7.9**	23.7	1.2	0.897
Buffer	0.003418	51.2	53.2	22.1	25.3	0.777
Rate	0.002478	57.3	18.5	17.3	37.5	0.869
FESTIVE	0.001368	26.7	18.0	**26.4**	**0.9**	0.404
Hybrid	0.005016	46.9	17.2	21.6	1.6	0.711
FairQ-smooth 15s	0.001269	66.5	11.8	24.4	10.1	1.008
FairQ 15s	0.001546	67.1	9.2	24.3	13.5	1.018
FairQ-smooth 20s	0.001680	65.4	11.7	24.2	10.1	0.992
FairQ 20s	0.001861	67.5	9.7	24.3	13.4	1.025
FairQ-smooth 25s	0.001326	66.4	11.8	24.3	9.8	1.008
FairQ 25s	**0.001038**	**67.6**	9.7	24.5	13.4	**1.026**

Table A.30: Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 1080p clients.

360p	Fairness	Util. (%)	Switches	Buffer (s)	I.P. Length (s)	Quality (Kbps)
Resolution	0.000716	15.0	**1.2**	27.9	1.4	0.228
Buffer	**0.000129**	**25.9**	2.1	27.7	13.1	**0.393**
Rate	0.000995	25.5	4.7	27.9	15.2	0.387
FESTIVE	0.002000	19.9	10.9	27.3	1.5	0.302
Hybrid	0.000923	21.2	7.0	23.7	**1.1**	0.322
FairQ-smooth 15s	0.002640	13.6	21.9	27.8	1.9	0.206
FairQ 15s	0.002074	14.3	15.6	27.8	1.6	0.217
FairQ-smooth 20s	0.001318	13.8	24.0	27.7	1.6	0.209
FairQ 20s	0.002418	13.9	16.7	27.8	1.7	0.212
FairQ-smooth 25s	0.001728	12.7	28.2	27.9	1.9	0.193
FairQ 25s	0.002301	13.2	16.8	**28.0**	1.8	0.200

Table A.31: Results from the Real Bandwidth Trace network scenario under a mixed client scenario showing only 360p clients.

Appendix B

Comparison of FairQ Variations by Metric

This appendix will compare the performance of each of the 6 variations of FairQ by the metrics used in Chapter 4 and 6 which are: (1) Fairness, (2) Utilization, (3) Quality Levels, (4) Quality Switches , (5) Buffer Levels, and (6) Initial Phase Length. We will create a section for each of these metrics and a table for each variation of FairQ under these sections. These tables will contain the results of that metric for the specific variation of FairQ across different client and network scenarios. These results are used in Chapter 6 to identify the best variation of FairQ under different client and network scenarios.

B.1 Fairness Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.001616	0.002470	0.000106	0.000198
Drop 2	0.001414	0.002425	0.000057	0.000133
Drop 3	0.000185	0.003217	0.004502	0.000091
Drop 4	0.000304	0.001052	0.001465	0.000132
Big Drop	0.000135	0.001639	0.002286	0.000242
Big Raise	0.001041	0.001171	0.000665	0.000222
Drop and Raise	0.001510	0.003240	0.004536	0.000013

Table B.1: Fairness produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.001370	0.002194	0.000009	0.000083
Drop 2	0.001920	0.002251	0.000000	0.000099
Drop 3	0.000515	0.001415	0.000855	0.000000
Drop 4	0.000623	0.003635	0.005068	0.000000
Big Drop	0.000263	0.004206	0.005899	0.000094
Big Raise	0.000474	0.002867	0.003235	0.000017
Drop and Raise	0.000828	0.003522	0.004739	0.000112

Table B.2: Fairness produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.002102	0.002562	0.000010	0.000215
Drop 2	0.002188	0.002557	0.000000	0.000215
Drop 3	0.000138	0.001252	0.001473	0.000020
Drop 4	0.000215	0.003246	0.004590	0.000000
Big Drop	0.000666	0.001282	0.001812	0.000041
Big Raise	0.000348	0.001622	0.001863	0.000000
Drop and Raise	0.001739	0.001127	0.001279	0.000131

Table B.3: Fairness produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.001543	0.002136	0.000009	0.000000
Drop 2	0.001929	0.002178	0.000044	0.000000
Drop 3	0.000522	0.003164	0.004088	0.000129
Drop 4	0.000657	0.003565	0.004927	0.000000
Big Drop	0.000407	0.001405	0.001758	0.000095
Big Raise	0.000949	0.001622	0.001363	0.000000
Drop and Raise	0.001791	0.001687	0.001384	0.000111

Table B.4: Fairness produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.001579	0.002446	0.000053	0.000000
Drop 2	0.000731	0.002460	0.000000	0.000099
Drop 3	0.000143	0.003208	0.004494	0.000111
Drop 4	0.000614	0.003687	0.005211	0.000202
Big Drop	0.000280	0.001262	0.001776	0.000020
Big Raise	0.000613	0.001479	0.001325	0.000091
Drop and Raise	0.001746	0.001183	0.001598	0.000255

Table B.5: Fairness produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	0.001970	0.002235	0.000000	0.000114
Drop 2	0.001332	0.002251	0.000017	0.000099
Drop 3	0.000413	0.001526	0.001301	0.000035
Drop 4	0.000610	0.001476	0.001768	0.000000
Big Drop	0.000182	0.003639	0.005098	0.000053
Big Raise	0.000501	0.002057	0.001227	0.000239
Drop and Raise	0.001546	0.003097	0.004121	0.000205

Table B.6: Fairness produced by FairQ 25s across network and client scenarios.

B.2 Utilization Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	79.4	79.7	72.9	6.8
Drop 2	79.5	79.8	73.0	6.8
Drop 3	81.6	81.9	74.6	7.4
Drop 4	82.0	82.3	75.0	7.4
Big Drop	80.4	80.6	73.3	7.2
Big Raise	79.2	79.2	72.0	7.2
Drop and Raise	79.4	78.0	71.0	7.0

Table B.7: Utilization produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	80.3	80.7	73.8	6.9
Drop 2	80.4	80.8	73.9	6.9
Drop 3	82.1	82.8	75.3	7.5
Drop 4	82.9	83.5	76.0	7.5
Big Drop	81.0	82.2	74.9	7.4
Big Raise	80.2	80.7	73.4	7.4
Drop and Raise	80.2	78.9	71.8	7.1

Table B.8: Utilization produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	79.3	80.0	73.1	6.8
Drop 2	79.3	79.9	73.1	6.8
Drop 3	81.5	81.4	74.1	7.3
Drop 4	81.7	82.2	74.9	7.3
Big Drop	80.9	81.0	73.8	7.2
Big Raise	78.8	79.7	72.5	7.2
Drop and Raise	79.8	77.4	70.4	7.0

Table B.9: Utilization produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	80.4	80.7	73.8	6.9
Drop 2	80.2	80.7	73.8	6.9
Drop 3	82.1	82.9	75.4	7.5
Drop 4	82.8	83.7	76.2	7.5
Big Drop	81.2	82.2	74.8	7.4
Big Raise	79.4	81.0	73.7	7.4
Drop and Raise	80.5	78.1	71.0	7.1

Table B.10: Utilization produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	79.3	79.9	73.1	6.8
Drop 2	79.5	79.8	73.0	6.8
Drop 3	81.7	81.6	74.3	7.3
Drop 4	82.1	82.5	75.2	7.3
Big Drop	80.3	80.9	73.6	7.2
Big Raise	78.7	79.4	72.2	7.2
Drop and Raise	79.5	78.0	71.0	7.0

Table B.11: Utilization produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	80.1	80.8	73.9	6.9
Drop 2	80.2	80.8	73.9	6.9
Drop 3	81.8	82.8	75.3	7.5
Drop 4	82.8	83.6	76.1	7.5
Big Drop	81.0	82.1	74.8	7.3
Big Raise	79.4	80.5	73.2	7.3
Drop and Raise	80.1	78.9	71.8	7.1

Table B.12: Utilization produced by FairQ 25s across network and client scenarios.

B.3 Quality Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.083	2.092	3.826	0.358
Drop 2	2.086	2.095	3.831	0.359
Drop 3	1.939	1.946	3.543	0.350
Drop 4	1.947	1.955	3.561	0.349
Big Drop	1.943	1.947	3.544	0.349
Big Raise	1.915	1.914	3.481	0.348
Drop and Raise	1.984	1.949	3.548	0.350

Table B.13: Quality levels produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.108	2.118	3.873	0.363
Drop 2	2.110	2.121	3.879	0.363
Drop 3	1.949	1.967	3.578	0.356
Drop 4	1.968	1.983	3.611	0.354
Big Drop	1.958	1.987	3.619	0.356
Big Raise	1.938	1.951	3.547	0.356
Drop and Raise	2.004	1.972	3.588	0.356

Table B.14: Quality levels produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.081	2.099	3.840	0.358
Drop 2	2.082	2.099	3.839	0.358
Drop 3	1.936	1.934	3.518	0.349
Drop 4	1.940	1.953	3.558	0.348
Big Drop	1.955	1.957	3.566	0.349
Big Raise	1.905	1.927	3.504	0.349
Drop and Raise	1.994	1.935	3.521	0.349

Table B.15: Quality levels produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.109	2.118	3.873	0.363
Drop 2	2.105	2.120	3.876	0.363
Drop 3	1.949	1.969	3.582	0.356
Drop 4	1.966	1.988	3.620	0.356
Big Drop	1.961	1.986	3.616	0.356
Big Raise	1.919	1.958	3.561	0.355
Drop and Raise	2.013	1.952	3.548	0.356

Table B.16: Quality levels produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.082	2.098	3.837	0.359
Drop 2	2.086	2.095	3.832	0.358
Drop 3	1.941	1.939	3.529	0.349
Drop 4	1.950	1.960	3.570	0.349
Big Drop	1.941	1.955	3.559	0.350
Big Raise	1.901	1.919	3.488	0.350
Drop and Raise	1.988	1.949	3.550	0.349

Table B.17: Quality levels produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.103	2.120	3.878	0.363
Drop 2	2.106	2.121	3.879	0.363
Drop 3	1.943	1.967	3.578	0.355
Drop 4	1.967	1.985	3.614	0.356
Big Drop	1.958	1.984	3.614	0.354
Big Raise	1.919	1.946	3.536	0.355
Drop and Raise	2.003	1.972	3.589	0.355

Table B.18: Quality levels produced by FairQ 25s across network and client scenarios.

B.4 Quality Switches Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	4.5	4.5	5.0	4.0
Drop 2	5.3	4.8	5.7	4.0
Drop 3	5.7	6.6	8.2	5.0
Drop 4	5.7	6.4	7.8	5.0
Big Drop	5.2	5.7	6.3	5.0
Big Raise	7.2	7.9	10.8	5.0
Drop and Raise	6.6	6.8	8.5	5.2

Table B.19: Quality switches produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	3.0	1.0	1.0	1.0
Drop 2	4.0	1.0	1.0	1.0
Drop 3	3.7	3.2	5.3	1.0
Drop 4	3.8	2.2	3.5	1.0
Big Drop	1.7	1.1	1.2	1.0
Big Raise	5.1	3.3	5.7	1.0
Drop and Raise	5.1	2.7	4.3	1.0

Table B.20: Quality switches produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	4.3	4.5	5.0	4.0
Drop 2	4.0	4.5	5.0	4.0
Drop 3	5.5	7.3	9.7	5.0
Drop 4	8.0	7.1	9.2	5.0
Big Drop	5.7	5.6	6.2	5.0
Big Raise	7.8	8.2	11.5	5.0
Drop and Raise	6.8	7.1	9.2	5.0

Table B.21: Quality switches produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.7	1.0	1.0	1.0
Drop 2	3.2	1.0	1.0	1.0
Drop 3	3.8	2.6	4.2	1.0
Drop 4	4.5	2.3	3.7	1.0
Big Drop	2.1	1.7	2.3	1.0
Big Raise	5.8	3.7	6.3	1.0
Drop and Raise	4.6	2.8	4.7	1.0

Table B.22: Quality switches produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	4.7	4.5	5.0	4.0
Drop 2	4.7	4.5	5.0	4.0
Drop 3	5.7	6.6	8.2	5.0
Drop 4	6.5	5.8	6.5	5.0
Big Drop	5.2	5.7	6.3	5.0
Big Raise	8.5	8.6	12.2	5.0
Drop and Raise	6.9	6.8	8.5	5.2

Table B.23: Quality switches produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	2.9	1.0	1.0	1.0
Drop 2	2.8	1.0	1.0	1.0
Drop 3	6.3	3.4	5.8	1.0
Drop 4	3.4	2.7	4.3	1.0
Big Drop	3.2	1.5	2.0	1.0
Big Raise	5.5	4.2	7.5	1.0
Drop and Raise	4.4	3.2	5.5	1.0

Table B.24: Quality switches produced by FairQ 25s across network and client scenarios.

B.5 Buffer Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	28.2	28.7	28.3	29.0
Drop 2	28.3	28.6	28.3	29.0
Drop 3	27.9	28.5	28.1	28.8
Drop 4	28.2	28.6	28.3	28.9
Big Drop	28.3	28.6	28.2	29.0
Big Raise	27.8	28.5	28.1	28.8
Drop and Raise	27.7	28.6	28.3	29.0

Table B.25: Buffer levels produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	27.9	28.6	28.1	29.0
Drop 2	27.8	28.4	27.8	28.9
Drop 3	27.8	28.3	27.8	28.8
Drop 4	27.8	28.3	27.6	28.9
Big Drop	28.2	28.5	28.1	29.0
Big Raise	27.3	28.2	27.5	28.9
Drop and Raise	27.3	28.5	28.1	28.9

Table B.26: Buffer levels produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	28.2	28.7	28.4	29.0
Drop 2	28.3	28.6	28.3	29.0
Drop 3	28.0	28.4	28.0	28.9
Drop 4	28.0	28.6	28.3	28.9
Big Drop	28.3	28.6	28.2	28.9
Big Raise	27.8	28.6	28.2	28.9
Drop and Raise	27.7	28.5	28.1	28.9

Table B.27: Buffer levels produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	27.9	28.5	28.1	29.0
Drop 2	27.9	28.5	28.0	28.9
Drop 3	27.8	28.2	27.6	28.8
Drop 4	27.7	28.2	27.6	28.9
Big Drop	28.2	28.5	28.1	28.9
Big Raise	27.4	28.2	27.5	28.9
Drop and Raise	27.5	28.5	28.0	28.9

Table B.28: Buffer levels produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	28.2	28.7	28.4	29.0
Drop 2	28.2	28.6	28.2	28.9
Drop 3	27.9	28.5	28.1	28.9
Drop 4	28.1	28.6	28.3	28.9
Big Drop	28.3	28.6	28.3	29.0
Big Raise	27.8	28.5	28.2	28.8
Drop and Raise	27.5	28.6	28.3	29.0

Table B.29: Buffer levels produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	28.0	28.4	27.9	28.9
Drop 2	28.1	28.4	27.8	28.9
Drop 3	27.8	28.2	27.7	28.8
Drop 4	27.8	28.3	27.7	28.9
Big Drop	28.2	28.5	28.1	28.9
Big Raise	27.7	28.1	27.3	28.9
Drop and Raise	27.6	28.4	28.0	28.9

Table B.30: Buffer levels produced by FairQ 25s across network and client scenarios.

B.6 Initial Phase Length Tables

FairQ-smooth 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.4	9.7	16.3	3.1
Drop 2	13.4	9.8	16.6	3.0
Drop 3	13.5	7.0	12.0	1.9
Drop 4	13.9	7.1	12.1	2.1
Big Drop	15.5	8.2	14.3	2.1
Big Raise	15.6	7.9	13.8	2.0
Drop and Raise	15.5	8.0	13.8	2.2

Table B.31: Initial phase lengths produced by FairQ-smooth 15s across network and client scenarios.

FairQ 15s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.4	9.6	16.3	2.9
Drop 2	13.6	9.6	16.3	2.9
Drop 3	13.5	7.1	12.4	1.8
Drop 4	13.9	6.9	12.0	1.7
Big Drop	15.7	8.4	14.3	2.4
Big Raise	16.3	8.3	14.5	2.1
Drop and Raise	15.8	8.2	14.4	2.0

Table B.32: Initial phase lengths produced by FairQ 15s across network and client scenarios.

FairQ-smooth 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.5	9.4	15.7	3.0
Drop 2	13.3	9.5	16.0	3.1
Drop 3	13.5	7.1	12.2	2.0
Drop 4	14.4	7.0	12.2	1.8
Big Drop	16.1	8.1	13.9	2.2
Big Raise	15.4	7.9	14.0	1.8
Drop and Raise	15.7	8.2	13.8	2.6

Table B.33: Initial phase lengths produced by FairQ-smooth 20s across network and client scenarios.

FairQ 20s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.2	9.8	16.4	3.2
Drop 2	13.6	9.9	16.5	3.2
Drop 3	13.6	7.1	12.5	1.8
Drop 4	13.9	7.0	12.2	1.8
Big Drop	15.9	8.3	14.2	2.5
Big Raise	15.8	8.3	14.6	2.1
Drop and Raise	15.8	8.6	15.3	1.9

Table B.34: Initial phase lengths produced by FairQ 20s across network and client scenarios.

FairQ-smooth 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.6	9.5	15.8	3.1
Drop 2	13.4	9.8	16.5	3.0
Drop 3	13.2	7.0	12.1	1.9
Drop 4	13.8	6.8	11.8	1.8
Big Drop	15.6	8.1	14.2	2.1
Big Raise	15.8	8.1	14.1	2.2
Drop and Raise	15.5	8.1	13.9	2.3

Table B.35: Initial phase lengths produced by FairQ-smooth 25s across network and client scenarios.

FairQ 25s	Homogeneous	Mixed	Homo in Mixed 1080p	Homo in Mixed 360p
Drop 1	13.5	9.4	16.1	2.8
Drop 2	13.8	9.9	16.4	3.4
Drop 3	13.4	7.3	12.4	2.2
Drop 4	13.6	7.1	12.4	1.8
Big Drop	15.4	8.5	14.4	2.6
Big Raise	15.8	8.3	14.2	2.3
Drop and Raise	16.0	8.2	14.1	2.2

Table B.36: Initial phase lengths produced by FairQ 25s across network and client scenarios.