2020

Network-Based Management for Optimising Video Delivery

Alissa, Ali Edan Taher

Copyright Statement

UNIVERSITY OF
PLYMOUTH

Network-Based Management for Optimising Video Delivery

by

Ali Edan Taher Alissa

A thesis submitted to the University of Plymouth

in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Electronics and Mathematics

May 2020

Acknowledgement

First and foremost, I express my sincere gratitude to the Almighty God for showering me with so many blessings.

Furthermore, throughout these four years, many scholars have advised me and provided invaluable assistance with my doctoral research. First and foremost, I would like to thank my Director of Study, Associate Professor Dr Bogdan Ghita, for his commitment and his excellent guidance. I would like to further extend my gratitude to my co-supervisors, Dr Is-Haka Mkwawa and Prof Thomas Zinner, for their excellent professional input in the process of conducting this research. Without their guidance and advice, this thesis could not have been completed, and I hope our collaboration can continue in the future.

My unreserved love, thanks and appreciation must go to my wife (Alaa) and children (Lujan and Tabarak), who have been very patient, understanding, and inspiring to me throughout this endeavour. I hope that the potential success of this research will compensate for some of what they have missed. May Allah bless them.

I would also like to thank my sisters, my brothers-in-law, my father-in-law, and my friends (Khaleel Odeh, Abdulrahman Alruban, Aissa Bouaissi, and Abdelhak Bentaleb) for their inspiration and support throughout my PhD journey.

I am also grateful to Al-Muthanna University and the Ministry of Higher Education and Scientific Research for granting me a scholarship and sponsoring my PhD studies.

Finally, I would like to dedicate this work to the soul of my first teacher (my father), Mr Edan Alissa, and my first love (my mother), Mrs Amal Hasan, who taught me the meaning of this life. Without their support, prayers, and inspiration, I would not be where I am today.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee. Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.
Relevant scientific seminars and conferences were regularly attended at which work was often presented:

Publications:

A. E. Al-Issa, A. Bentaleb, T. Zinner, I. Mkwawa and B. Ghita, "BBGDASH: A Max-Min Bounded Bitrate Guidance for SDN Enabled Adaptive Video Streaming," 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2019, pp. 307-314, doi: 10.1109/ICIN.2019.8685894.

A. E. Al-Issa, A. Bentaleb, A. A. Barakabitze, T. Zinner and B. Ghita, "Bandwidth Prediction Schemes for Defining Bitrate Levels in SDN-enabled Adaptive Streaming," 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 2019, pp. 1-7, doi: 10.23919/CNSM46954.2019.9012713.

Word count for the main body of this thesis: **41816**

Signed:

Date: 08/05/2020

Dedication

To my father
who took the lead to heaven
before the completion of this work

Abstract

Network-Based Management for Optimising Video Delivery

Ali Edan Taher Alissa

The past decade has witnessed a massive increase in Internet video traffic. The Cisco Visual Forecast index indicates that, by 2022, (79%) of the world's mobile data traffic will be video traffic. In order to increase the video streaming market revenue, service providers need to provide services to the end-users characterised by high Quality of Experience (QoE). However, delivering good-quality video services is a very challenging task due to the stringent constraints related to bandwidth and latency requirements in video streaming. Among the available streaming services, HTTP adaptive streaming (HAS) has become the de facto standard for multimedia delivery over the Internet. HAS is a pull-based approach, since the video player at the client side is responsible for adapting the requested video based on the estimated network conditions. Furthermore, HAS can traverse any firewall or proxy server that lets through standard HTTP data traffic over content delivery networks. Despite the great benefits HAS solutions bring, they also face challenges relating to quality fluctuations when they compete for a shared link. To overcome these issues, the network and video providers must exchange information and cooperate. In this context, Software Defined Networking (SDN) is an emerging technology used to deploy such architecture by providing centralised control for efficient and flexible network management. One of the first problems addressed in this thesis is that of providing QoE-level fairness for the competing HAS players and efficient resource allocation for the available network resources. This has been achieved by presenting a dynamic programming-based algorithm, based on the concept of Max-Min fairness, to provide QoE-level fairness among the competing HAS players. In order to deploy the proposed algorithm, an SDN-based architecture has been presented, named BBGDASH, that leverages the flexibility of the SDN's management and control to deploy the proposed algorithm on the application and the network level. Another question answered by this thesis is that of how the proposed guidance approach maintains a balance between stability and scalability. To answer this question, a

scalable guidance mechanism has been presented that provides guidance to the client without moving the entire control logic to an additional entity or relying purely on the client-side decision. To do so, the guidance scheme provides each client with the optimal bitrate levels to adapt the requested bitrate within the provided levels. Although the proposed BGGDASH can improve the QoE within a wired network, deploying it in a wireless network environment could result in sub-optimal decisions being made due to the high level of fluctuations in the wireless environment. In order to cope with this issue, two time series-based forecasting approaches have been presented to identify the optimal set of bitrate levels for each client based on the network conditions. Additionally, the implementation of the BBGDASH architecture has been extended by proposing an intelligent streaming architecture (named BBGDASH+). Finally, in order to evaluate the feasibility of deploying the bounding bitrate guidance with different network conditions, it has been evaluated under different network conditions to provide generic evaluations. The results show that the proposed algorithms can significantly improve the end-users QoE compared to other compared approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the past decade, video traffic has witnessed a massive increase within the Internet traffic. According to Cisco Visual Forecast index [4], (79%) of the world's data traffic will be video traffic by 2022. Moreover, the majority of video traffic originates from different devices, such as smartphones, TVs, computers, etc. Furthermore, different technologies can be adopted to provide streaming services, such as Video on Demand (VoD), P2P video streaming, Augmented Reality (AR) and Virtual Reality (VR) video streaming. These factors are anticipated to further increase the video streaming market in future networks. In 2018, the global video streaming market size was valued at USD 36.64 billion and is expected to continue to grow at a CAGR of 19.6% from 2019 to 2025. The popularity of online video streaming, including media such as Netflix and YouTube, are also expected to rise in future networks. The rapid adoption of smartphones also contributes to the expansion and popularity of social media platforms for the digital market. The growing adoption of cloud-based video streaming solutions that increase the reach of video content is also contributing to the growth of the video streaming market.

In order to increase the video streaming market revenue, service providers need to provide services that meet the end-user's requirements and utilise the available resources. Furthermore, the concept of the Quality of Experience (QoE) has been introduced as a user-centric metric for evaluating the perceived quality or services. However, achieving good QoE is a highly challenging task; this is because of the

stringent constraints related to end-user requirements, the services, and the unstable nature of the underlying network infrastructures. To present a real-case scenario, different users can use heterogeneous devices to receive videos within heterogeneous environments such as stadiums, shopping malls, and trains, where limited resources have to be shared among the different users.

Generally speaking, VOD services are released using two streaming methods: (1) IP Protocol Television (IPTV), and (2) Over-The-Top (OTT). For many years, IPTV has been an appealing technology for use by service providers to offer video content to their clients over the Internet. Furthermore, in the IPTV approach, the service providers utilise their own network infrastructure to deliver video content. Therefore, the challenge in IPTV systems from the service provider perspective is that of how to build a suitable network delivery architecture that exploits the network resources and reduces costs. It is worth mentioning that, with IPTV, the Service Level Agreement (SLA) and Experience Level Agreement (ELA) are guaranteed based on the network's Quality of Service (QoS). On the other hand, within OTT services, the contents are delivered via the best-effort delivery (i.e. the Internet) without a dedicated network infrastructure; this makes the video delivery more challenging, as the content provider does not manage both the network infrastructure and the end-user devices.

Moreover, different protocols (e.g., HTTP, RTP, RTSP and UDP) are used within OTT services at the transport and application layers. In the 1990s and early 2000s, most videos were delivered to users using real-time transport protocols (RTP) [135]. RTP runs over UDP to deliver the media data over the IP-based Internet. It was developed by the Internet Engineering Task Force (IETF) as an IP-based standard protocol for real-time video streaming services. In parallel with RTP, Real-Time Control Protocol (RTCP) [135] is used to control the delivery of RTP's video streaming sessions. Furthermore, within RTP/RTCP, video content is typically pushed by the media server to each user, which could result in scalability issues occurring at the server-side. Nevertheless, RTP has some other limitations, including the following: (a) it can be affected by traversal issues (for firewalls and NATs); (b) it requires a customised strategy for content caching rather than the traditional web-caching.

Unlike the RTP protocol, HTTP Adaptive Streaming (HAS) [129] is capable of

traversing any firewall or proxy server that lets through standard HTTP data traffic over content delivery networks. HAS is a pull-based approach, which allows players to adapt the requested bitrate level based on the estimated network conditions. In this way, HAS eliminates the scalability issues on the server-side, helps to achieve the highest possible video quality, and ensures better network resource utilisation. This has caused it to be adopted by the most popular video service providers such as Apple HTTP Live Streaming [3], Microsoft Smooth Streaming [13], and Adobe HTTP Dynamic Streaming [1] for both live and on-demand video streaming.

1.2 Research Challenge

This section presents the research challenges regarding HAS that are addressed in this thesis. Despite the benefits that HAS brings to video streaming, recent studies [121], [125], [85], [116] have shown that many widely used adaptive players suffer from multiple performance issues, including unfairness in bandwidth allocation among multiple players, network resource underutilisation, video quality switching and instability, or video freezes [121]. The main objectives of the HAS algorithms are to maintain the possible highest video quality and to maximise network resource utilisation. Delivering good video quality to heterogeneous clients that share the same bottleneck is in fact a more challenging task compared to a scenario involving a single player. This is the case because different HAS players have different requirements (i.e. screen size, media formats, etc.) and they are normally based the network-level fairness to share the available resources. By contrast, the latter cannot guarantee QoE level fairness.

In HAS, the video segments are stored in the server so that the player on the client's side can retrieve them. During video streaming, it is challenging for the player to avoid mismatch from occurring between selected video bitrate and real network throughput. Such a mismatch can be the result of variations in encoding bitrate, the uncertainty of the throughput measurements, or the limitation of the available video representations, which leads the client to stream with a bitrate level that does not match the real network throughput. Furthermore, the player may encounter continuous playback interruptions because of sudden congestion. In order

to cope with video freezes, the playback buffer must be maintained at a sufficient level. To do this, the adaptation logic must move between different video bitrates in order to retrieve a reliable level and keep a certain level of the video buffer[123]. Nevertheless, Mok *et.al* [99] indicate that streaming a video at a higher average bitrate does not always guarantee higher QoE to the end users. This is because frequent switching between the different video levels to cope with sudden network bandwidth variations may lead to degradation in end-user QoE Seufert *et.al* [121]; therefore, the adaptation strategy should consider the impact of quality fluctuations within the design of the adaptation algorithm.

Today, there are various devices used for video streaming that have different capabilities and features, such as screen size, storage and processing power, and streaming videos over heterogeneous networks (3G/4G/LTE). This means that the only factor in selecting video segments from the server is the adaptation logic that adapts the quality locally at the client side [44]. Therefore, designing an algorithm that can deliver video to the end users via various devices is a highly challenging task. It is worth noting that the designed mechanisms should meet the end users' requirements and adapt to the network conditions. Furthermore, accurate measurement of the available network resources is critical in order to adapt the quality efficiently. However, different locations have varying network conditions; therefore, the strategy for measuring the available network resources should be adaptable to the dynamic of the network conditions.

End players may request videos with different requirements (i.e. video resolution, bitrate levels, etc.). Allocating the available resources equally may therefore cause unfairness in terms of network resources that have to be shared among users. The existence of different DASH clients during video streaming, all of which are competing for limited network resources on a shared bottleneck link, may lead to the following problems [37], [58]: (a) unfair distribution of the available resources, (b) bandwidth underutilisation, and (c) video instability.

The root cause of this unfairness among the different players is the traffic pattern (ON-OFF), which is generated during the download of the video segments [24]. In more detail, if the segment download time is less than the segment playout time, then after the start-up phase, and when the buffer reaches a certain level, the client be-

comes idle (i.e. OFF period); the client then enters the ON period (i.e. downloading the video segment) when the buffer level drops below the maximum level. Furthermore, bandwidth underutilisation during video streaming prevents the clients from utilising the available resources and requesting a higher bitrate level. Therefore, the proposed bitrate adaptation strategy should eliminate the impact of the ON-OFF traffic pattern and provide efficient network resource utilisation.

Both academia and industry have made numerous attempts to improve video delivery while providing higher QoE to the end users. Some of these solutions have considered enhancing the video quality based on purely client-based HAS approaches, which are in turn based on the local estimation (i.e. throughput, buffer level) for the bitrate adaptation [121]. Other solutions are based on network-based assistance, in which a network component is utilised to allocate the available network resources among DASH players.

Purely client-based HAS solutions suffer from quality fluctuations when they compete for a shared link [24]. Furthermore, other QoE-related metrics, such as the features of the requested video, are not considered within the ABR algorithms, which might result in an unfair distribution of video quality among different users. To overcome these issues, network and video providers must exchange information and cooperate. Emerging technologies, such as the server and network-assisted delivery (SAND) architecture [130], offer standard signalling for the network-to-client and network-to-network communication of quality-relevant information. This architecture allows the network elements to apply the appropriate policies that match network conditions and user requirements. In this context, Software-Defined Networking (SDN) [79] is an emerging technology used to deploy such architecture by providing centralised control for efficient and flexible network management. However, moving the entire bitrate adaptation engine into an external entity (i.e. network-based component) harms the principle underpinning the pull-based bitrate adaptation approach and leads to scalability issues. To this end, a new bitrate adaptation strategy is required that eliminates the above-mentioned issues affecting the purely client-based and fully network-based approaches.

1.3 Research Aims and Objectives

From an analysis of the above-mentioned challenges, it may be concluded that there is a need to explore more solutions for improving video streaming services and utilising the available resources. These proposed solutions should consider the requirements and capabilities of the end-user devices. Furthermore, the proposed solutions should maintain the principle of DASH at the end client. Nevertheless, the suggested adaptation schemes need to be able to adapt to the conditions of the underlying network infrastructure. In this regard, this thesis focuses on addressing the following technical questions:

- How can we provide QoE-level fairness for the competing HAS players and efficient resource allocation for the available network resources?
  To answer this question, a dynamic programming-based algorithm based on the concept of Max-Min fairness is presented to provide QoE-level fairness for the competing HAS players and efficient resource allocation for the available network resources.

- How should the proposed guidance algorithm be deployed?
  In order to deploy the proposed algorithm, an SDN-based architecture named BBGDASH is presented that leverages the flexibility of the SDN management and control to deploy the proposed algorithm on both the application and the network level.

- How can the proposed guidance approach maintain a balance between stability and scalability?
  To answer this question, a scalable guidance mechanism is presented that guides the client without either moving the entire control logic to an additional entity or relying purely on client-side decisions.

- How should the proposed QoE guidance approach be extended to work within a wireless environment?

  As wireless network conditions are subject to considerable fluctuations , providing efficient QoE guidance is a non-trivial issue, as a significant QoE degra-

dation could result if the client is guided with an incorrect set of bitrate levels. Therefore, in order to deal with this issue, two time series-based forecasting approaches are presented that identify the optimal set of bitrate levels for each client based on the network conditions. Additionally, the implementation of the BBGDASH architecture is extended by proposing an intelligent streaming architecture (referred to as BBGDASH^{+}).

1.4 Thesis Structure

This thesis is organised as follows. Chapter 2 begins with an overview of Internet video streaming technologies over the past three decades, including HTTP adaptive streaming. It then summarises the concept of Quality of Experience (QoE), including its measurement and optimisation. The chapter goes on to outline the concept of Software-Defined Networking (SDN), with a particular focus on its design and applications, and ends with a brief overview of time series forecasting methods.

Chapter 3 presents an extensive review of proposed approaches for optimising the delivery of adaptive video streaming, including client-based approaches, network-based approaches, and server-based approaches.

Chapter 4 presents a novel and scalable network-assisted approach, called Bitrate-bounded Guidance DASH mechanism (hereafter BBGDASH), which identifies the boundary range of the requested bitrate levels while preserving the final quality adaptation for the client. The chapter also summarises the implementation of a network-assisted video streaming framework that leverages the functionality of SDN where BBGDASH is deployed. In addition, it also presents a use-case-based evaluation that demonstrates the feasibility and the potential of the proposed approach to deliver video over SDN-enabled networks while providing high QoE to the end-users.

Chapter 5 presents two time series-based forecasting approaches that identify the optimal set of bitrate levels for each client based on the network conditions. Additionally, this chapter extends the implementation of the architecture presented in chapter 4 to an intelligent streaming architecture (referred to as BBGDASH^{+}). Furthermore, the chapter includes a set of experimental evaluations with different configuration parameters to investigate the behaviour of the proposed approaches

under real network conditions. Finally, Chapter 6 provides conclusions, limitations and future work.

Chapter 2

Background

2.1 Internet-based Video Streaming

This chapter presents a brief overview of video streaming technologies. It also outlines the diverse range of applications and transport protocols, such as HTTP, RTSP, RTP, TCP/STCP and UDP, which have been adopted by major service providers for video streaming.

2.1.1 History of Internet-based video streaming

Over the last decade, the biggest growth area in Internet usage has been video streaming. Users over the past few years have exhibited increasing interest in viewing video content and watching movies on-demand using different devices, such as desktop computers, tablets, laptops, smartphones, and televisions. Furthermore, Augmented Reality (AR) and Virtual Reality are emerging video technologies that promise to transform many industries and alter the way they work. However, deploying such technologies may result in a tremendous increase in Internet usage. In order to cope with the increasing demands of video, telecommunications networks require higher transmission speed and improved network design.

Within traditional non-HAS video streaming, video content is delivered to the end-users using one of the two main paradigms [2]: (a) connection-oriented protocols (e.g., Real-time Messaging Protocol (RTMP/TCP)), or (b) connectionless protocols (e.g., Real-time Transport Protocol (RTP/UDP)). Moreover, Real-time Streaming Protocol (RTSP) [120] is a protocol that has been used to control the transmission

of video from media servers to clients in multimedia streaming networks. During video streaming, RTSP first conducts the streaming session setup and maintains the state information throughout this streaming session, while protocols such as RTP are responsible for the delivery of video content from the server to the client. It is worth mentioning that RTP is session-oriented, in that it provides data for the application to perform (a) identification for the source and the payload type, (b) detection of the lost packets, (c) synchronisation between video playout and jitter buffer, and (d) synchronisation between audio and video during a streaming session. During video streaming, RTSP augments the RTP by periodically transmitting the control packets. It also provides feedback about the delivered data and statistics about session participants.

In the early 1990s, there were several players that were able to play videos over the Internet. The video server was responsible for delivering the video content to clients through unicast (i.e. one-to-one) connections [90]. Note that each unicast client connecting to the video server uses its bandwidth. For example, five clients watching a 4Mbps video stream use a total of 20Mbps bandwidth, while if only one client is watching the video, the network traffic used is 4Mbps. For this reason, deploying the unicast streaming paradigm becomes extremely bandwidth-consuming when the network supports a large number of clients and a large number of video streams from the server. In this case, the network infrastructure must be able to handle this scenario based on the video traffic and user demands. The scalability issue of the unicast approach for video streaming prompted proposals for new IP multicast protocols, which are more scalable when streaming a video to a large number of users. IP multicast is commonly used for Internet Protocol television (IPTV) applications and the delivery of content in both corporate and private networks [90]. IP multicast systems require both the network and client devices to support Internet Group Management Protocol (IGMP) snooping. Figure 2.1 presents an illustration of unicast, multicast and broadcast streaming to different numbers of users.

The main challenge for video multicasting today is the heterogeneity of the devices being used, which have different capabilities, especially when it comes to delivering QoE to users, as multicasting for video streaming cannot provide a video bitrate that is appropriate for different hardware features and network transmission

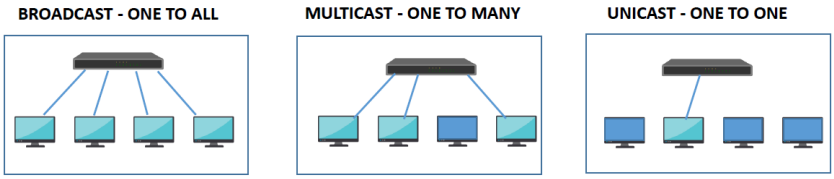


Figure 2.1

Digital Video Distribution over IP networks

capacities of multiple users. Another form of video streaming is through broadcast, where a video server transmits each video stream to all devices connected to the network, as shown in Figure 2.1.

In light of the above-mentioned issues associated with traditional non-HAS streaming, the focus of academia and industry has been directed toward an alternative solution that can be deployed on top of HTTP. The first implementation of video streaming over HTTP/TCP protocols was called HTTP progressive download. During video progressive downloading, the clients must request a file through an HTTP GET. The video file is then sent to the client using the HTTP protocol. It is worth mentioning that this approach is similar to how web objects such as pictures, images and text files are downloaded from a regular web server using the standard web browser. With progressive downloading, only a small portion of the video file needs to be downloaded before playback begins. This implies that the video is actually downloaded to the client, and that the video starts to play when it is available on the local computer. Note that, if the user needs to fast forward or skip to another video section of the same streaming video, then the user can do that only if a portion of the video has already been delivered and stored on their machine. In order to download a portion of the video file, the progressive video downloading client has to access the Web server. The client then plays out the buffered data while it is downloading future portions of the same video. Some of the advantages of progressive video downloading include the following: (a) it can be deployed on the top of traditional web servers; (b) its friendliness with firewall and NAT devices; and (c) it supports CDN because of the massive popularity of HTTP.

Unfortunately, deploying the HTTP progressive download comes with a number of drawbacks. The main issue is that the client has to manually choose the video

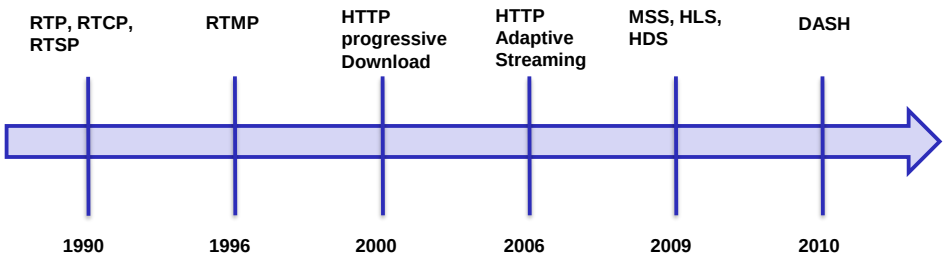


Figure 2.2

History of Video Streaming

quality at the start of the video streaming. This manual mechanism leads to play-out interruptions, especially in highly variable network conditions. One solution for this issue is to keep a sufficiently large buffer size; however, this solution could waste network resources every time the client switches to another video. The drawbacks of the progressive download have pushed the streaming community to find an alternative streaming paradigm, which is represented by HTTP adaptive streaming.

HTTP Adaptive Streaming (HAS) was first introduced by Move Networks in 2007 [89]. Within HAS, the contents of the video are encoded into multiple bitrate levels and chopped into small video segments, each of which has a typical duration of two to 12 seconds. Furthermore, an adaptation mechanism runs at the client side that dynamically adapts the bitrate level of the requested segment based on the network throughput. Adopting this approach helps to reduce the video interruption during video streaming and enables improving the perceived QoE at the client-side compared to the progressive download approach discussed before. HAS quickly acquired the attention of the dominant video streaming companies and has been deployed widely; for example, Adobe released Adobe HTTP Dynamic Streaming (HDS) [1], Apple released Apple HTTP Live Streaming (HLS) [3] and Microsoft released Microsoft Smooth Streaming (MSS) [12]. Despite the popularity of HAS, however, different implementations of HAS are not compatible. This incompatibility between the different deployments of HAS pushed the Universal Mobile Telecommunications Systems – Long Term Evolution (UMTS-LTE) to release the first standard of HAS in 2009. Further extension through collaboration with MPEG ended with the standardisation of MPEG-DASH (i.e. DASH) in 2012 [11], after which the second edition of MPEG-DASH was released in 2014. The DASH standard defines

guidelines for content presentation, segmentation, and delivery. However, the adaptation logic is not standardised, and it is left open for academia and industry to explore and find novel solutions. Apart from standardisation bodies, the DASH Industry Forum (DASH-IF) [6] is another body that has been promoting the adoption of MPEG-DASH since 2012. DASH-IF is made up of major streaming and media companies, including Netflix, Microsoft, Ericsson, Google, Samsung, Adobe and many others. DASH-IF is also responsible for shaping the success of MPEG DASH through providing guidelines and recommendations on the usage of MPEG DASH.

2.2 Dynamic Adaptive Streaming over HTTP (DASH) framework

Figure 2.3 presents the general DASH architecture, which consists of the DASH server, the DASH clients, and the communication protocols. On the server side, videos are encoded into multiple levels that are usually identified by the video bitrate. Each level is generated by compressing/encoding the raw video with one of the standard codecs (i.e. H.264, H.265/HEVC, VP9, etc.). Generally speaking, video bitrate is proportional to video quality; however, this relation between the video quality and its bitrate is not always linear. There are two main approaches used for encoding the video bitrate, namely the constant bitrate (CBR) and the variable bitrate (VBR). Under the CBR encoding schema, the video is compressed with a constant level of bitrate and quality levels that vary along with the video. The other schema keeps the quality constant at the cost of bitrate variability. Furthermore, videos on the server side are segmented into small chunks of equal duration (i.e. two to 12 sec.) using one of the DASH segmenting tools, such as GPAC MP4Box [9]. In order to provide seamless streaming and smooth switching between the different bitrates, each chunk is independently encoded. Therefore, each segment begins with an Intra-coded frame (I-frame) that makes the segment independent of other segments.

As shown in Figure 2.4, each DASH video on the server-side is coupled with its metadata file, referred to as the Media Presentation Description (MPD). MPD is an

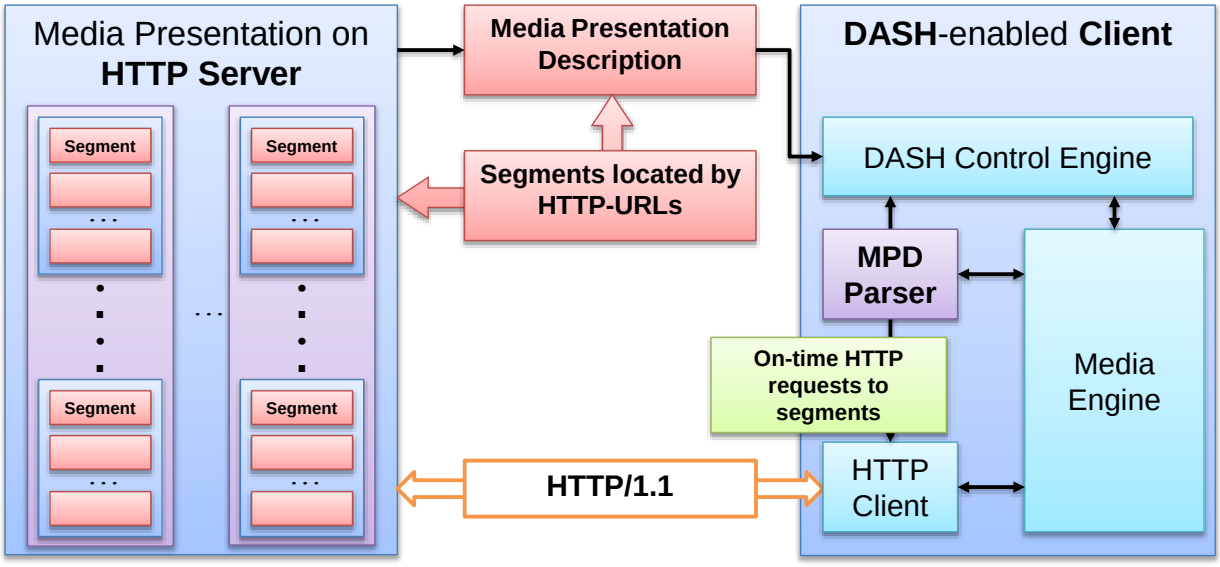


Figure 2.3

DASH Architecture [132]

Extensible Mark-up Language (XML) file that hosts all content-related information (i.e. video bitrates, resolutions, URL location of the video chunks, video codecs, etc.). Furthermore, this information is hierarchically structured within the MPD file, with the period as the root element of the XML file. Each period represents a video object with a specific duration along with one or more adaptation sets that all versions of the content, each of which is referred to as a representation. Each representation hosts the initialisation segment that contains the metadata of the representation along with other video segments. For example, Figure 2.5 shows the implementation of the MPD file.

Figure 2.6 illustrates the behaviour of the DASH system when delivering video content. As shown, each DASH client starts by requesting the MPD file from the server side. Upon receiving the MPD file, the client issues a set of HTTP requests (typically in chronological order) for the video segments. All downloaded segments are stored in the video playback buffer (buffering phase) before the video starts playing (playing phase). Furthermore, each playback buffer (video buffer) has a maximum size (i.e. up to 30 seconds). Therefore, it is often the case that the client has ON/OFF download patterns, especially when the arrival time for the requested segment is less than the playback time. For instance, if the requested segment has

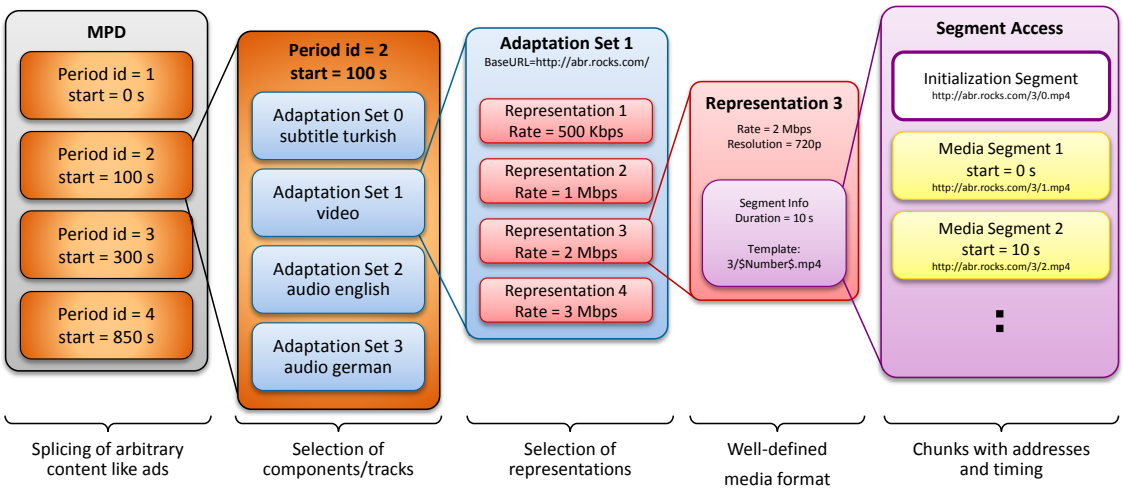


Figure 2.4

DASH Media Presentation Description (MPD) [132]

```
<AdaptationSet segmentAlignment="true" group="1" maxWidth="480" maxHeight="360" maxFrameRate="24" par="4:3"
 subsegmentStartsWithSAP="1">
  <Representation id="320x240 46.0kbps" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height="240"
  frameRate="24" sar="1:1" startWithSAP="1" bandwidth="45652">
    <BaseURL>bunny_45652bps/BigBuckBunny_2snonSeg.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="885-4504">
      <Initialization range="0-884" />
    </SegmentBase>
  </Representation>
        .
        .
        .
        .
        .
```

Figure 2.5

Example of the Media Presentation Description (MPD)

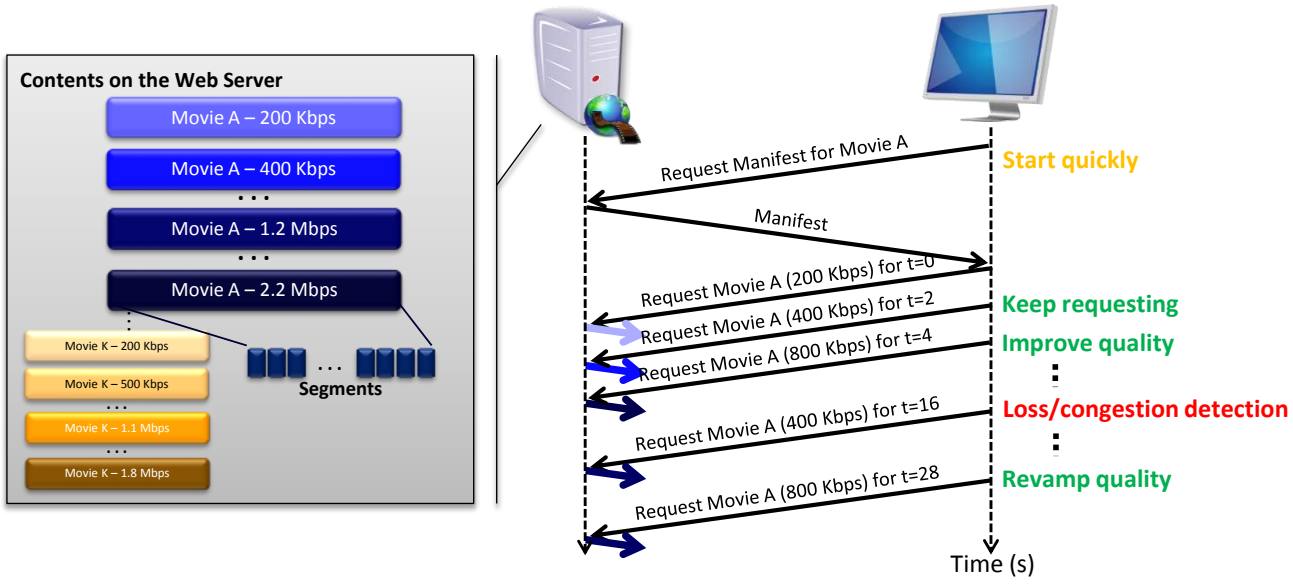


Figure 2.6

Example of client-centric HTTP Adaptive Streaming session [132]

video bitrate levels of 2, 4 and 6 Mbps, while the network throughput is 5 Mbps, then the buffer grows more quickly or more slowly than the playing time according to the selected bitrate level. In order to provide smooth playback streaming and select the most appropriate segment at the right time, each DASH player is equipped with a local adaptive bitrate algorithm (ABR) that adapts the quality and requests the bitrate level that matches the network conditions. Different approaches to ABR algorithms have been presented in the literature, with most of them deployed at the client side. Most of the client-side bitrate adaptation approaches are based on the segment download time and/or playback buffer occupancy level to estimate the network conditions and request the segment that best fits the estimated network conditions.

Most of the bitrate adaptation algorithms have been built to run under HTTP/1.1, in which every segment is delivered under the HTTP request/response pattern. However, HTTP/2 [80] has been deployed recently to act as the application protocol within the DASH architecture. HTTP/2 offers a suite of features, including server push; accordingly, under HTTP/2, the video server is able to send more than one segment for each client request. Consequently, HTTP/2 has the potential to enhance the delivery of DASH traffic, especially with live streaming. Other ben-

efits that can be gained from deploying the HTTP protocol on top of the DASH architecture are as follows: (1) DASH allows the client to dynamically adapt the requested bitrate level according to the network conditions; (2) DASH traffic can traverse easily through NATs and firewalls; (3) content providers can utilise conventional Web servers for hosting and cashing DASH content; (4) DASH servers are mostly stateless, as each DASH player requests the segment independently, and this behaviour allows the clients to request videos from multiple servers and hence balances the load among servers; (5) DASH can improve system scalability, as it does not require a persistent connection between the client and the server.

2.3 Quality of Experience (QoE): Concepts and Definition

Initially, the Quality of Service (QoS) concept was used to assess video quality for Internet-based streaming. The QoS concept has been considered as an indicator for estimating the quality of the services transmitted over the computer network. However, QoS is based only on the network parameters (i.e. packet loss, jitter, etc.) for evaluating the quality of the received services without considering the other QoE-related factors. Accordingly, it does not reflect the quality of the services received by video consumers in the end-to-end aspect of audio-visual streaming systems.

In order to overcome the limitations of the QoS, the user-centric approach (i.e. QoE) [46], [20] was proposed in 2007 and presented as a user-centric approach for measuring perceived quality. The QoE-centric concept is comprehensively defined in [46] as follows:

'The degree of delight or annoyance of the user of an application or service. It results from the fulfilment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state, expectation or perception'. Unlike QoS, which considers the traditional network metric toward the measurement of the quality of the measured services, QoE takes other contextual factors into account to evaluate the service quality. Fig 2.7 illustrates the QoE concept, where an end user is subjected to perceiving the video quality from YouTube, or quality while watching a movie. In the context of future

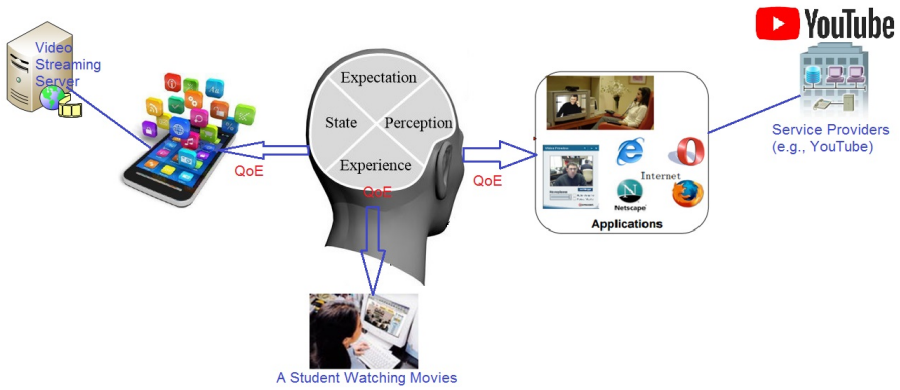


Figure 2.7

Illustration of the QoE Concept

networks, QoE is considered as the main indicator for measuring the perceived quality of the multimedia services.

2.3.1 Quality of Experience Influence Factors

QoE covers all factors in the end-to-end communication system (e.g., from service providers or operators to clients). These factors can play a significant role in influencing the user's experience of the delivered service. As shown in Fig 2.8, these factors can be divided into three categories based on the influence factor (IF): namely, System IFs, Context IFS and Human IFs [43]. System Influence Factors (SIFs) are features that define the technically produced quality of service. SIFs can be further classified into four subcategories: namely, content-related, media-related, network-related and device-related SIFs. In the video streaming scenario, the type of video (i.e. sport, anime, etc.) corresponds to content-related SIFs, while the bitrate, encoding, and resolution configuration parameters are related to media-related SIFs. Network and devices capabilities are related to the network-related and device-related SIFs, respectively [43].

Context IFs refer to all factors that comprise characteristics of the user's environment. These factors include temporal features (e.g. the time of day that the video is played), the physical characteristics (e.g. the location of the played video), and the economic aspects (in term of the cost of the played video or the subscription type of the end-client). Human IFs are related to the characteristics and the properties

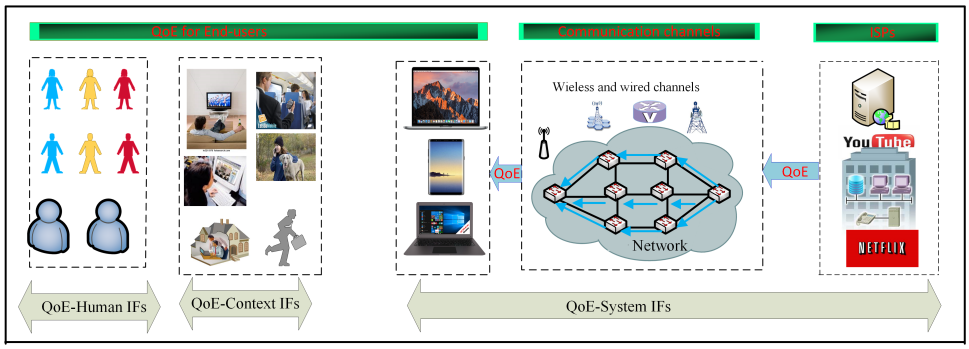


Figure 2.8

Illustration of IFs for QoE provisioning in the context of future networks

of the end-user, as different users may perceive the quality differently according to their emotional and mental temperament [114].

The influence factors used for QoE assessment of video streaming services can be measured objectively and quantifiably. However, although these factors can be used to assess video quality, they cannot accurately reflect QoE. The next subsection provides a discussion of some QoE assessment methods related to subjective tests, objective quality models and data-driven analysis models for quantifying the end-users' video quality.

Subjective Quality Models

Subjective video quality reflects the quality received by the end user. Participants assess video quality using a Mean Opinion Score (i.e. MOS) on a scale of 1 (bad) to 5 (excellent). In order to conduct the subjective test, the original videos are watched under various conditions to generate the processed video sequences, which are in turn assessed by the end-user according to the MOS index. Some of the reference methods for conducting subjective video quality assessment are provided by the International Telecommunication Union (ITU) [19]. The plans for performing subjective tests in the laboratory are provided and guided by the Video Quality Expert Group (VQEG) [18]. The VQEG engages in collaborative efforts to improve subjective video quality test methods, as well as the developing and validating of objective video quality metrics. For the industry, the VQEG seeks to improve the understanding of new video technologies and applications. Although MOS studies have served as the basis for analysing many aspects of video quality, they also

present several limitations: (1) subjective tests for assessing the video quality require stringent environments with limited participants/subjects, test videos and testing conditions; (2) the process cannot be automated; (3) it is very costly and time-consuming to repeat this process frequently; and (4) it is impossible to use these procedures for real-time quality assessment.

Objective Quality Models

Objective quality models for assessing video quality were introduced to address the limitations of subjective test methods. Most of the objective methods tests mathematically compare the source and encoded files and deliver a score for each tested video. Note that different objective methods focus on how the Human Visual System (HVS) processes and perceives the video signals. The most commonly used method involves quantifying the physical difference between the reference and target (distorted) video. The errors are then weighted according to spatial and video temporal features of the video

According to the available information between the original and the delivered video content, the objective models can be categorised into three groups: full-reference (FR), reduced-reference (RR) and no-reference (NR). The first category (i.e. FR) measures the perceived quality of the received video by comparing the frames of the original and the delivered videos. There are multiple video metrics, including peak signal-to-noise ratio (PSNR), structured similarity index (SSIM), video quality matrix (VQM), and SSIMPlus. Intuitively, the utility of each metric relates to its ability to predict how human eyes would evaluate the files or the correlation with subjective results. The Mean Squared Error (MSE) is the average squared difference between the two signals (i.e. the distorted and reference signals). In other words, MSE is a signal fidelity measure, which provides a quantitative score of the similarity or fidelity of two signals. The Peak Signal-to-Noise Ratio (PSNR) is widely used to measure the difference in quality between the distorted and the reference signal. These models are commonly used in a range of studies, such as [40] and [41], and usually serve as the benchmark for assessing video quality. The PSNR-HVS is an extension of PSNR that incorporates properties of the human visual system, such as contrast perception. This metric [133] is a modification of

PSNR that utilises the contrast sensitivity function and a coefficient based on the HVS. The model inputs are the original 8x8 pixel block and the corresponding block of the distorted image. The DCT of the difference between the given pixels is calculated and then reduced by the value of contrast masking [133]. This method can be used for both overlapping and non-overlapping blocks. According to the available information between the original and the delivered video content, the objective models can be categorised into three groups: full-reference (FR), reduced-reference (RR) and no-reference (NR). The first category (i.e. FR) measures the perceived quality of the received video by comparing the frames of the original and the delivered videos. There are multiple video metrics, including peak signal-to-noise ratio (PSNR), structured similarity index (SSIM), video quality matrix (VQM), and SSIMPlus. Intuitively, the utility of each metric relates to its ability to predict how human eyes would evaluate the files or the correlation with subjective results. The Mean Squared Error (MSE) is the average squared difference between the two signals (i.e. the distorted and reference signals). In other words, MSE is a signal fidelity measure, which provides a quantitative score of the similarity or fidelity of two signals. The Peak Signal-to-Noise Ratio (PSNR) is widely used to measure the difference in quality between the distorted and the reference signal. These models are commonly used in a range of studies, such as [40] and [41], and usually serve as the benchmark for assessing video quality. The PSNR-HVS is an extension of PSNR that incorporates properties of the human visual system, such as contrast perception. This metric [133] is a modification of PSNR that utilises the contrast sensitivity function and a coefficient based on the HVS. The model inputs are the original 8x8 pixel block and the corresponding block of the distorted image. The DCT of the difference between the given pixels is calculated and then reduced by the value of contrast masking [133]. This method can be used for both overlapping and non-overlapping blocks.

The Structural SIMilarity (SSIM) index indicates the similarity level between the two images. It identifies the quality measure of one of the images when compared with the other perfect quality image [139]. As an extension of SSIM, SSIMplus is an objective full-reference perceptual video QoE index with a range between 0 and 100. In many ways, SSIMplus goes far beyond what SSIM can measure. The

SSIMplus distinctive features include: (1) high accuracy and high speed; (2) being straightforward and easy to use; (3) providing device-adaptive and cross-resolution QoE assessment; (4) providing cross-content QoE measurement and detailed quality mapping during video assessment. SSIMplus may be employed in many application scenarios. For example, in the field of video delivery over multimedia communication networks, SSIMplus may be applied in the following ways: (1) live and file-based video QoE monitoring; (2) benchmarking video encoders and transcoders; (3) guiding adaptive bit-rate video coding; and (4) enabling smart quality-driven adaptive bit-rate video streaming.

The Video Quality Metric (VQM) [109] is a collection of several tools that can be used to measure the quality of a video signal. This is a full-reference, signal-based metric. VQM is standardised under ITU-T J.144 [112] as the NTIA General Model. The VQM metric aligns the frames between the reference and distorted video with compensation for frame losses (called spatial and temporal registration). It then calculates different features in the spatiotemporal domain and compares them between the reference and distorted video.

The Video Multimethod Assessment Fusion (VMAF) index is a fusion-based perceptual video quality metric, proposed by Netflix in collaboration with the University of Southern California, which predicts video quality by fusing multiple elementary metrics using machine learning algorithms [102]. The advantage of fusing multiple elementary metrics in this way is that it allows the strengths of the individual metrics to be used while overcoming their inherent weaknesses. The current version of the VMAF algorithm and model, called VMAF 0.3.1, uses Support Vector Machine (SVM) regression. The software for the VMAF algorithm is available as part of the VMAF Development Kit (VDK), which also offers tools for training and testing custom VMAF models [101].

The Video Intrinsic Integrity and Distortion Evaluation Oracle (VIIDEO) [98] is a blind video quality assessment (VQA) model that requires no information about the video quality. VIIDEO can perform quality prediction of distorted videos without any external knowledge about the training videos containing anticipated distortions, or human opinions of video quality or original video source. Although VIIDEO seems to perform better than existing blind IQA models [97], [98], [88],

it may fail to represent some video-specific intrinsic characteristics due to the fact that it can only capture the common baseline characteristics of a specific piece of content.

Despite the efficiency of the FR models in providing a good quality estimation, deploying such models does require the presence of the source and the delivered video. Accordingly, significant efforts have been made toward developing new models that can estimate the perceived quality of the received video without the need to assess the reference video. These models assess the received quality either through leveraging some features of the original video (i.e. RR) [82] [138] or utilising the network and application statistics, such as packet loss, jitter, latency, bitrate levels, etc (i.e. NR) [143] [136] [75] [117].

Data-driven Analysis Models

The amount of data associated with different content types sent over the Internet has increased in recent years. Different data with different quality metrics, along with user engagement during video streaming, has also been increasing on the Internet. As a result, data-driven analysis has been recognised as an approach capable of covering some of the limitations of objective and subjective methods when it comes to measuring and estimating the end user's QoE. The data-driven analysis incorporates user engagement metrics [122], [32] such as the number of videos watched, the viewing time, return rate and abandonment rate. The QoE indicators such as rebuffering, video bitrate, stall duration and startup delay, all of which are measurable and quantifiable, are normally used to derive the user engagement metrics used during data-driven analysis. The data-driven analysis approach can be used to develop simple but efficient QoE prediction models. This is primarily made possible through the use of big data and machine learning paradigms, which are able to learn from previous data and forecast the end user's video bitrate and quality requirements. Content providers are interested in data-driven analysis methods, since they enable their business objectives and revenue generation targets to be easily met.

2.3.2 Quality of Experience for Adaptive Bitrate Streaming (ABS)

Internet-based streaming employs ABS to deliver videos to the end-users. For many years, the QoE topic has been attracting the attention of the multimedia community among both academia and industry. The main driving factor for this is that different factors (e.g., startup delay, number of stalls and their duration, latency, bitrate switching etc.) contribute to delivering video with excellent quality to the end users. The following subsection provides a description of each factor that influences QoE delivery to users.

Playback Stalls or Rebuffering Events

Playback stalls during video streaming may occur when the downloaded segment fails to arrive before its playback deadline. This occurs when the video buffer at the client side is empty. This streaming situation, also referred to as 'buffer underflow', is a condition that can generate buffering events where the video data is accumulated on the client's device in order to resume the video playback. In some streaming scenarios, the rebuffering or playback stalls can be treated as the initial delay (i.e. the waiting time before the requested video starts working). Despite that, these two events have different impacts on the end-user's perceived QoE.

Several works in the literature indicate that playback stalls or rebuffering are significant for QoE measurement and assessment [60]. For example, the authors in [54] show that rebuffering has the most significant impact on user engagement, while other studies show that frequent-but-shorter stalls [100] are worse than fewer-but-longer stalls for video streaming [61].

Video Bitrate and Switching Impacts

QoE is also influenced by the video quality, which is usually characterised by the bitrate. The achieved video bitrate level influences the overall QoE by affecting the video quality. Therefore, selecting the highest possible video quality of segments is one of the essential rules for maximising the end-user's QoE. In no HAS traditional progressive video streaming, the main factors that effecting perceived QoE are initial delay, video stalling, and the average video quality [60]. While the adaptive

behaviour of HAS during the video streaming results in adding a new additional influence factor (i.e. quality fluctuation) on the perceived QoE [104]. Nevertheless, the quality fluctuation is represented by video switching frequency (i.e. number of video switching events during the video session) and video switching amplitude (i.e. the bitrate/quality difference between video switching events). Authors in [146] showed that quality fluctuation may have a negative or positive impact base on the direction of video switching events (i.e. downgrade/upgrade quality level). However, the impact of the downward quality change on the end-user QoE is stronger than the upward quality switching. Another work in [99] shows that an intermediate video quality switching has a lower negative impact than sudden video quality downgrade.

Startup Delay

The startup delay refers to the time from the moment at which the video is requested until the start of the video being played back on the client device. This also includes the time required to download the other corresponding files, such as the manifest file (the MPD file) in MPEG-DASH. In an adaptive video streaming scenario, the end client's playback buffer has to store a certain amount of data before video playout. The startup delay to be achieved during video streaming must take the trade-off between the buffering sizes into account; a higher startup delay is the result of a larger buffering size.

This also provides the DASH client with more robustness against buffer underflow under highly variable network conditions. Furthermore, buffering size is heavily affected by video parameters such as the video bitrates and segment duration, as well as network impairments (e.g., network delay and throughput). In the literature, several studies have investigated the effects of the initial delay on QoE. Authors in [87] showed that as the startup delay increases, the QoE decreases; however, the negative impact of the buffer underflow on QoE is stronger than the initial delay. Another work, [115], uses an exponentially decaying function to model the relationship between the QoE and the startup delay, while [128] find that within IPTV and VoD services, startup delays can be tolerated if the reward is less video stalling during streaming.

Latency

In the case of live streaming, live latency is another factor considered during the QoE evaluation process. Furthermore, in a HAS-based live streaming scenario, the end-to-end latency includes content encoding, content segmentation, initial delay, and buffering time. However, a VoD streaming service does not have the same latency requirements as live services. This is because, for VoD, the video content is already available for streaming to the client's device, while live streaming services require interactivity and active participation (such as augmented vision, online gaming and video conferences; [137]).

Furthermore, it is worth mentioning that the user-level QoE of DASH media streams depends on two subjective factors, namely:

- The temporal video quality
- The spatial video quality

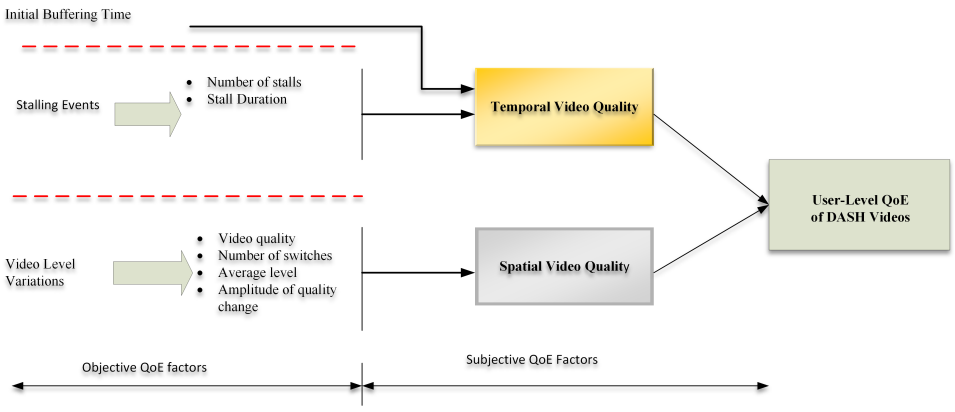


Figure 2.9
User-Level QoE metrics for DASH videos [86]

As shown in Fig 2.9, the temporal quality of the perceived video is determined by the initial buffering time and stalling events, while the spatial video quality is determined by the video-level variation. Consequently, all QoE metrics (i.e., video quality, initial buffering time, number of bitrate switching, and stall duration) are considered to provide an accurate measurement for the perceived QoE of media streams at the end user.

2.3.3 Maximising the QoE for Adaptive Video Streaming

The primary goal of ABS is to provide an excellent QoE level at the end user level. However, achieving such a high QoE level is non-trivial, as it requires achieving a balance between different QoE factors that may conflict with each other during the adaptation process. The main goals of ABS are listed below:

- Eliminating the video stalling events, which occurs when the video buffer on the client side runs out of data.
- Maximising the possible bitrate level of the playing video, which is associated with vastly more video stalling events when the video is streamed under highly variable network conditions.
- Reducing the initial delay of the video start-up time that happens when the client starts streaming at a high bitrate level.
- Mitigating the frequent and persistent video quality switching that occurs when the adaptation logic is highly sensitive to the network conditions.

Examining the above goals of ABS, it can be seen that there are some conflicts between them. In order to achieve the first aim and stream video smoothly, the requested video bitrate level needs to be lowered; however, this could result in sub-optimal video quality. Second, maximising the video bitrate level could result in a high number of video stalling events when the networking conditions are unstable; the conflict is obvious between these two goals. Furthermore, the initial start-up delay can be mitigated by starting with a lower bitrate level; again, however, this could lead to reduce the perceived QoE. Furthermore, providing a smooth quality level with a minimal number of video switching events boosts the perceived QoE at the end user. However, at highly variable network conditions, fixing the quality level leads to more video freezes during a video streaming session. Nevertheless, modelling QoE for HAS-based streaming remains an open question, as each model weights these factors differently [33]. Therefore, the above challenges need to be considered when designing a robust ABR algorithm that achieves the intended goals.

2.4 Fairness

Network fairness is another important metric that needs to be considered when designing any resource allocation system. Internet traffic mostly relies on the TCP protocol to achieve fair allocation of the available resources. However, many studies [25] [70] have revealed the limitation of the adaptive video streaming algorithms in terms of their ability to provide fair network resource allocation when they share a common bottleneck. Furthermore, in a typical video streaming scenario, different DASH players may request different videos for which the maximum bitrate levels are not the same. Therefore, enforcing the same bitrate allocation for all users may result in an unfair distribution and inefficiencies in both QoE and network resource allocation. Meanwhile, max-min fairness is a feasible fairness measure that maximises the minimum video flow, then allocates the remaining resources among other video flows. Max-min fairness was thus considered within the design of the optimisation algorithm deployed within the proposed network-based system.

2.5 Software-Defined Networking

2.5.1 Traditional Network Infrastructure Model and Challenges

The traditional network has evolved over many decades and has been proven to be useful in different service provider environments. The traditional network infrastructure is based on the proprietary network devices, such as switches and routers, while the physical connections (i.e. wired and wireless) are used to connect the network components. The main role of the network components is to forward the network traffic from one point to another. However, these components have limited knowledge of the network as a whole.

Nevertheless, the Internet has witnessed tremendous growth in network data traffic over the past decades, along with an increasing number of different access networks (e.g., LAN, WLAN, Wifi, 2G/3G/4G/LTE etc.). Furthermore, various Internet-based service providers are facing increasing demands from their clients. The ability of the traditional network to adapt to business demands has also been

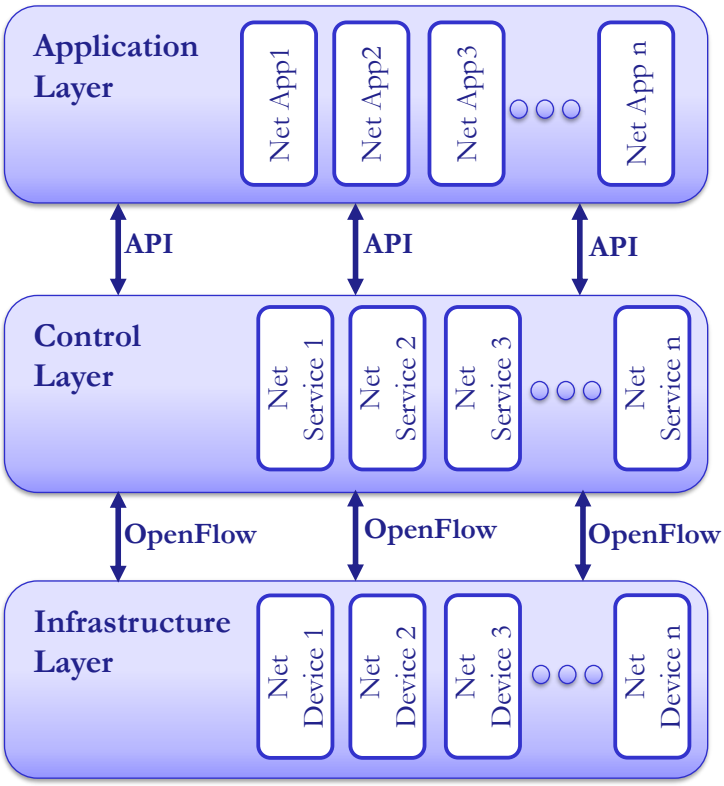


Figure 2.10

An overview of SDN Architecture

another concern in recent years. The growing requirements of service providers have made traditional networks unable to offer the required services to their customers. This is because the control and management within a traditional network are done manually by the system administrator, who typically configures devices when problems occur within the network.

Software-Defined Networking (SDN) [79] is a promising technology that aims to transform the networking architecture by decoupling the decision-making entity (the control plane) from the underlying network forwarding system (the data plane). SDN provides a centralised view of the entire network through a central controller. Moreover, SDN offers a mechanism that supports the control and management of physical and virtual resources, along with all networking devices, such as virtual switches using a controller. The next subsection presents the benefits, the design, and the challenges of SDN.

2.5.2 SDN Design

Within SDN, several software engineering concepts are applied, including modularity, abstraction, and reusability; this is done to enable the developer to develop flexible, scalable, and applicable applications that can efficiently manage future networks and meet the increasing demands of different internet-based services. Nevertheless, SDN provides network programmability in such a way that running software can control network infrastructure with no regard for network hardware. The high-level language will then be translated into hardware-based specifications through an open programming interface between the control plane and the data plane. Finally, the network components (i.e. forwarding devices) implement the network actions based on the received rules.

Figure 2.10 illustrates the SDN architecture, which consists of three layers: namely, the Infrastructure Plane Layer (Data Layer), the Control Layer, and the Application Layer. Furthermore, the Northbound and Southbound Interfaces are used to connect the Control Layer with the Application Layer and the Data Layer respectively. The descriptions of each layer are presented below.

Data Layer

The data layer, also known as the forwarding layer, represents the network infrastructure in which a set of network components are interconnected with other components using wired connections or common wireless radio channels. These network components can be switches, routers, firewalls, or any other network device. Every single network device in the data plane can have one or more interfaces, which are used to form a route of communication among devices. Nevertheless, the main functions of this layer are to forward the network traffic based on the received rules of the top layers and report the network statistics. Open VSwitch (OVS) is a good example for the virtual switches. This is a virtual switch designed for Hypervisors based on the Linux Operating system, which also supports different SDN-based protocols, including OpenFlow (OF) [95] and Open-VSwitch Data Base Management Protocol (OVSDB). Nevertheless, the Southbound Interface (SBI) provides the communication interface between the data layer and the control layer. While it offers multiple protocols, OF is the protocol most commonly used to control and manage

communications between the control layer and the data layer.

Control Layer

The control layer (also referred to as the controller) represents the operating system of the network components, as it instinctively indicates an intermediary layer between the data plane layer and the application layer via open programming interfaces. It also interprets the application requirements into hardware-level language, which in turn simplifies the development of applications. Furthermore, this layer acts as the 'brains' of the network, as it controls the network operations and provides the other layer (i.e. the application layer) with the information it requires. The control layer consists of an SDN controller used to monitor the underlying network resources and control the behaviour of the network elements based on the requirements of the application layer. The controller such as Floodlight and OpenDaylight enables communications to the network applications through the REST APIs. It also communicates with network devices (e.g., OpenFlow switches and routers) in the data forwarding layer via the SBI enabled with a standard protocol such as OpenFlow. Features such as network traffic control, programmability, automation and high performance are provided and enabled by the OpenFlow standard.

Application Layer

The application layer represents the top layer within the SDN architecture. This layer consists of a set of programs that reflect the requirements of the network stakeholders. Furthermore, it normally communicates with third-party applications and translates the received policies into network-based strategies. These applications are responsible for making decisions pertaining to the end-user demands and changing network conditions. It is worth mentioning here that the scope of applications in SDN include (a) load balancing, (b) traffic engineering, (c) network management and monitoring, (d) enforcement of QoS/QoE policy, (e) QoE management, and (f) enforcement of security and access policies, among others. The northbound interface (NBI), such as RESTful APIs, is used to provide communication between the control layer and the application layer.

OpenFlow Standard

The Open Networking Foundation (ONF), as the foundation of SDN technology, also provides the OpenFlow standard for enabling communication between different layers in the network. As an open standard, OpenFlow defines the communication between data forwarding devices (such as OpenFlow switches and routers) and the SDN control layer. In this way, it allows the decoupling of the data plane and control plane in SDN. This also enables higher functionalities and programmability features within the SDN networking approach.

OpenFlow protocol is an open standard that was developed by an open consortium and is currently supported by different networking and services provider companies for the implementation of various solutions related to load balancing, security, QoS/QoE management and monitoring, as well as resource orchestration. A network switch in an SDN-based network is responsible for storing the forwarding rules that originate from the SDN controller. After receiving this rule or policy, the OpenFlow switch makes this forwarding decision based on the port number from which this packet is coming. It is worth noting that the SDN controller is the one that takes control of the forwarding rules and their computation within an SDN network. This behaviour enables OpenFlow switches to maintain only minimal information when making forwarding decisions during normal SDN networking operations. In this way, it allows OpenFlow switches to have minimum memory and CPU requirements for processing data in SDN. Figure 2.11 illustrates the OpenFlow architecture, which consists of switches, Flow Table, Group Table and Meter Table.

As shown in Figure 2.11, an OpenFlow switch consists of one or more OpenFlow channels that connect directly to an external SDN controller. This controller is responsible for managing the switches through the OpenFlow protocol. It is important to note that one of the main benefits offered by the OpenFlow protocol is that it allows for the interoperability of switches produced by different vendors or manufacturers. This is possible even when these devices contain different interfaces and other proprietary functions. Due to this openness, OpenFlow can facilitate the programming of switches and remotely perform configuration of forwarding rules and various actions in OpenFlow tables. Nevertheless, an OpenFlow switch is logically made up of one or more flow tables and group tables; these tables are responsible

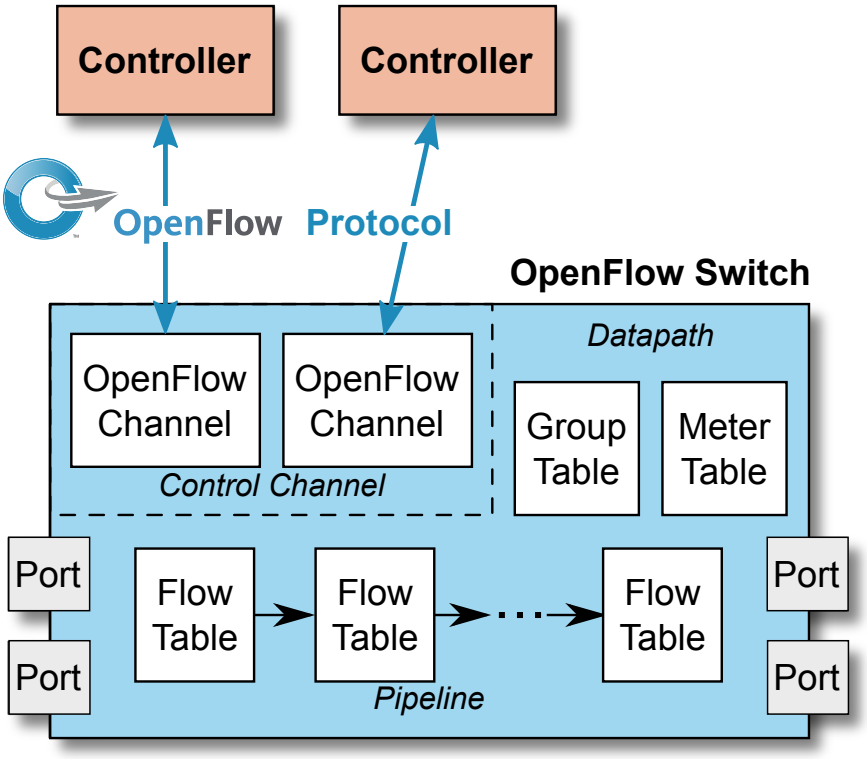


Figure 2.11

An OpenFlOW Architecture [15]

for performing table lookups and forwarding decisions.

Flow Table

The flow table represents the packet forwarding table in the OpenFlow switch. It consists of different sets of flow entries, which comprise different packet fields that are responsible for matching forwarding rules and actions as defined in the switch by the controller. For example, the controller can receive a message regarding an incoming packet, which is unknown, from a packet forwarding device. When the rules are known, the forwarding device can use existing rules and action plans that have already been set in the flow entry of the flow table to perform actions on the data packet.

Group Table

In the course of flow forwarding, the incoming flow can be forwarded into a group table instead of the flow table; the latter (i.e. flow table) enables OpenFlow to forward the incoming packets through multiple ports. This new feature introduces port group abstraction, which allows different forwarding behaviours in SDN related to link aggregation and the multipath routing of data packets. Moreover, the group table in an OpenFlow switch consists of different group entries. Every entry consists of a different list of action buckets that carry specific semantics of different group types. These group types of each group table can be specified as select, fast failover, and indirect.

Meter Table

The meter table is responsible for providing the QoS mechanisms in SDN. It achieves this by applying the flow rate of data packets. It also continuously monitors the rate of data packets prior to output. The flow entry in this table is called a meter. Each meter is directly attached to a flow entry in the table of an OpenFlow switch. A meter can measure, control and monitor the rate of network traffic defined by a specific flow in real time. By using the goto-meter action, data packets in SDN can be directed from the flow table's entry to a meter. The meter can then perform different operations on the data packet in order to limit the flow rate or combine with QoS queues to implement different QoS mechanisms in order to control and manage the different services.

2.5.3 SDN Benefits

SDN network management provides many and varied benefits [79], including its simplicity, automation, global network perspective, and optimal network management. These benefits have promoted SDN to become a network innovation enabler. The next subsections discuss the primary advantages of the SDN.

Simplicity and Convergence

SDN reduces the complexity of hardware programming and makes it easier to deploy network applications that manage the network elements and monitor the underlying network elements. This simplicity reduces the cost and time required by the software development cycle, and all requirements from the developer are up to code with the available SDN interface. Furthermore, the SDN convergence and openness enhances application collaboration and operations upgrades from different vendor's technologies that are centred on the same SDN concept.

Automation

SDN facilitates network programmability and automatability, which in turn enable the control plane to manipulate, control, and change the network strategy depending on the network conditions. Moreover, these network policies can be applied without any need for manual intervention. This feature also reduces operational errors that could arise due to human error.

Global Knowledge

SDN provides a centralised view of the entire network through a central controller. The latter enables SDN to monitor and collect the network stats and events. Consequently, based on the collected network stats, the control layer forms a holistic view of the network conditions, which can be fed as an input for the network applications to enable them to adapt their strategy.

Network Efficiency

SDN offers automated network traffic management using a centralised controller, which makes it easier to manage the available network resources and implement a network management strategy that optimises the network-based services. This ensures that the end users receive the best possible services, as the network responsiveness is greatly enhanced and the available network resources are efficiently utilised.

On the basis of the above benefits, SDN is one of the most appropriate technologies to employ when focusing on QoS/QoE monitoring and management. Indeed,

it has been experiencing a continuous evolution regarding the types of multimedia services with which the customers are provided and the way these are deployed; accordingly, new QoE prediction models for the monitoring and management of limited available resources are required. With SDN, such an evolution can be followed by feeding the control units with the new bandwidth prediction/forecasting models, which then guide the probing activities at the data forwarding nodes and terminals exploiting the softwarisation of the SDN process. In principle, this approach seems to have great potential. However, extensive efforts are needed to define appropriate techniques and demonstrate their effectiveness through experiments.

2.5.4 SDN Challenges

The academia and industry have identified SDN as an appealing solution for the management and control of resources and monitoring of network events. However, there are many challenging issues that can negatively affect the performance of the SDN in the context of actual implementation. These challenges include (1) scalability, (2) consistency, (3) flow limitations, (4) interoperability, and (5) security. The first three of these have been considered within the design of the proposed solutions, as explained in the following subsections.

Scalability

The centralised nature of SDN raises a number of scalability and performance issues, which can in turn affect the performance of the network. For example, the SDN controller can be a connection bottleneck when dealing with an amount of data that it cannot handle. Further, the reactive nature of the OpenFlow protocol also can cause a burden to the SDN controller since new rules have to be installed and redirected by the controller in different network components. This issue has been considered carefully within the design of the proposed solutions that aim for optimising the running services (i.e. video streaming) without moving all the task into the controller side.

Consistency

Consistency is another challenge faced by SDN, as the SDN controller changes the behaviours of the network components by updating the rules. The network components confirm the execution of the installed rules by notifying their controller. However, this behaviour could lead to network consistency in cases where the network elements confirm the applied update without executing that update. This issue has therefore been also considered within this work, as the controller has a partial role in achieving the required services rather than relying entirely on the SDN controller.

Flow Limitations

The current version of the OF protocol has multiple issues related to flow operations. The OF protocol presents a unidirectional flow rule, which defines the transmission from the source to the destination. Therefore, another rule needs to be installed to facilitate bidirectional communication. However, as the number of applications and clients increases, more flows need to be installed, which could result in a very large table; this will increase the lookup time required for finding the match flow in the flow table, which in turn decreases the network performance.

2.6 Time Series Analysis and Forecasting

2.6.1 Time Series Concept

The term 'time series' refers to a sequential set of data points, each of which is characterised by two mandatory components: namely, the time unit and the value corresponding to that point [22]. Each value may have one or more variables (i.e. univariate or multi-variate). Nevertheless, time series can be expressed mathematically as a set of chronological order measurements:

$$X(t), where \;\; t = 0, 1, 2, ..., n \tag{2.1}$$

where t is the time elapsed. Furthermore, time series measurements can be categorised based on how the time series data are recorded. For the first type (i.e. discrete), the time series measurements are collected at discrete points in time; this

may occur periodically or occasionally depending on the case being measured. In all cases, however, a discrete set of values are generated, which is why this type is formally denoted as a discrete time series. In the second type (i.e. continuous), the observations are collected continually along with the time intervals. For example, periodic network throughput measurement, temperature measurement, etc. can be considered as instances of a continuous time series. On the other hand, the population difference between two cities or the exchange rate between two different currencies are good examples of discrete time series.

Time series can be further classified into multiple categories. The first category is based on the distance between the consecutive observation values. This feature is used to group the time series into equidistant and non-equidistant time series. In the former class, the time series values are collected with a constant length of time between observations, while the latter (i.e. non- equidistant) does not maintain a fixed distance between the consecutive observations. Another category is based on the dependency between the consecutive time-series observations, which classifies time-series into long memory and short memory time series. Long memory time series are differentiated with slowly decremented autocorrelation functions such as network throughput measurement for the local area network, while short memory series are those with high decremented autocorrelation functions. Stationarity is another time-series classification factor that classifies series into stationary and non-stationary series. The former, unlike the latter, is identified with unique statistical characteristics in which the series has a constant mean over a constant time. In order to provide an accurate forecasting process, the time-series data need to be in stationary form.

2.6.2 Time Series components

According to Diggle [53], time series comprise four systematic components that can be detected from the observed time series: trends, cyclical, seasonal, and random noise. The trend component refers to the time series mean movement over a long period of time, which can be either linear or non-linear. For example, a series related to the number of internet users may exhibit an upward trend, while a downward trend can be observed in a series related to the rate of death in the UK. The seasonal

component refers to the fluctuations of the series within a year. Seasonality is influenced by different factors, such as season, climate, etc.; for example, power consumption increases in winter, while sales of fizzy drinks increase in summer. The cyclical component is another fluctuation that may occur within the series, and is mostly caused because of changes in circumstances. Finally, time series may have unpredictable influences that do not exhibit a regular pattern; these kinds of fluctuations are referred to as random noise.

Based on the interactions between the four time series components, two mathematical models (i.e. the additive model and multiplicative model) can be used to explain how these four components interact.

$$\text{Additive model} \qquad X(t) = T(t) + S(t) + C(t) + E(t) \tag{2.2}$$

$$\text{Multiplicative model} \qquad X(t) = T(t) * S(t) * C(t) * E(t) \tag{2.3}$$

Here, X(t) denotes the time series, while T(t), S(t), C(t), E(t) represent trend, seasonal, cyclical, and error respectively. Furthermore, it is important to mention here that with the additive model, the four components are assumed to be independent, while the multiplicative model assumes dependency between the four components.

2.6.3 Time Series Analysis

As presented in the previous section, time series are extremely applicable in different sectors for use in managing resources based on predicted scenarios. Traffic prediction for the available network resources is one of these applications. The network operators use prediction to manage and control the load of their networks. However, proper analysis is essential in order to understand and identify the components of the given time series data. Therefore, this section explains the aims of the time series analysis and the processes involved. Time series analysis refers to different methods that are applied to the given time series data to identify the relevant information. There are two main objectives of time series analysis:

1. Interpreting and understanding the nature of the time series data by identifying the main components that adequately describe the behaviour of the time

series.

2. Time series forecasting, in which the future values of the time series data are predicted based on the previous observations.

In order to achieve the aforementioned aims, different statistical methods are applied to identify the main time series components (i.e. trends, seasonality, cyclical behaviour, and irregular changes). Defining these components helps to construct the hypothetical stochastic representation of the data. Such representation is referred to as a model. The next step is to estimate the corresponding model parameters for the given time series data. The generated model then can be used to predict the future values through applying the previous value of the developed model.

2.6.4 Time Series Forecasting

Time series forecasting can be defined as the set of techniques used to predict the future values of a given time series based on the observed features. It can be applied across different fields of study, from economics to geology to network traffic prediction. The general idea is based on the concept of applying one of the time series methods to the given series to create a mathematical model that represents this series. The forecasting model will be used to predict future values based on the observed data. It is important to emphasise the difference between a forecasting method and a forecasting model. The former represents the set of algorithmic actions that are applied to the given time series to create the forecasting model. Based on the applied forecasting method, the way of measuring the forecasting accuracy can be identified. On the other hand, the forecasting model represents the mathematical representation of the given series, which can be used to predict future values of the time series.

Time series forecasting can also be categorised based on the forecasting task. Within the first category, the future values of the given series are predicted based on the computation of the same series, while the other category considers external factors in the computation of the time series forecasting.

2.6.5 Forecasting Accuracy

Forecasting accuracy is a metric that represents the performance of the forecasting model. Forecasting accuracy is the opposite of forecasting error, which measures the degree of closeness between the predicted data and the real data of the time series:

$$E_t = Y_t - F_t \tag{2.4}$$

Here, E is the forecasting error at period t, Y represents the real value at period t, and F is the predicted value at period t.

When evaluating the accuracy of the forecasting model and the goodness of the model parameters, forecast error can be used. Among the most common forecasting error metrics are root mean square error (RMSE) and mean absolute percentage error (MAPE). For a given number of observations N, the RMSE expresses the squared error between the the predicted F_t and the observed value(s) for the same measurement levels. MAPE is used to assess the predictions between different datasets. Both metrics can be described mathematically as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(Y_t - F_t)^2} \tag{2.5}$$

$$MAPE = \frac{100}{N} \times \sum_{t=1}^{N} \mid \frac{(Y_t - F_t)}{Y_t} \mid \tag{2.6}$$

2.6.6 Summary

In this chapter, the history of internet-based video streaming is presented, followed by presenting the underlying concept and components of Dynamic Adaptive Streaming over HTTP (DASH). Next, the concept of Quality of Experience (QoE) for the different video streaming approaches is outlined in detail, including the influence factors and the ways of maximising the received QoE. The design, benefits and challenges of the Software-Defined Network (SDN) are then discussed. Finally, the time series forecasting and analysis are presented.

Chapter 3

Literature Review

3.1 Introduction

This chapter presents the state-of-art quality adaptation approaches that have been proposed in the literature. The use of HAS for over-the-top (OTT) video services has prompted researchers to develop new adaptation approaches that optimise the perceived QoE by dynamically adapting the requested quality depending on the network conditions. It also drove network operators and service providers to present novel schemes that utilise the available resources and maximise the total revenue. Most existing research contributions are classified according to the location of the adaptation engine. There are three main categories of bitrate adaptation schemes: client-based, in-network-based, and server-based. Nevertheless, the design of the adaptation algorithm is not fixed, and is instead flexible in selecting where the adaptation algorithm should be deployed and on which metric(s) it should be based.

3.2 QoE-driven Adaptive Video Streaming: State of the Art

In this section, the most important video adaptation schemes in the literature are presented. Furthermore, the adaptation approaches can be classified based on where the adaptation logic is deployed; therefore, as discussed above, existing works can be grouped into three main categories: the client-based, server-based, and in-network based approaches [33].

3.2.1 Client-based Rate Adaptation

Most of the adaptation algorithms presented in the literature are deployed on the client-side, where each algorithm adapts the requested bitrate based on the estimated network conditions. Accordingly, every ABR employs one or more feedback signal(s) (i.e. throughput, buffer level, etc.) to estimate the network conditions. Depending on these feedback signals, client-side approaches can be broadly classified into four categories: throughput-based, buffer-based, artificial-intelligence-based and hybrid-based approaches. The following sections present the most important algorithms in each category.

Throughput-based Rate Adaptations

This category of algorithms adapts their quality locally by relying on the estimated throughput of previously downloaded segments. However, the network conditions vary over time, and HAS runs on the top of the TCP protocol, which is quite sensitive to fluctuation. Therefore, deploying such an approach may result in an inaccurate estimation unless these defects are taken into consideration. Further, as HAS is segment-based streaming, then the estimation is performed on the segment(s) level. This estimation can be typically calculated by dividing the size of the requested segment(s) by the time required to download it.

One of the early works of this type was by Liu et al. [44], in which the authors proposed a method that uses a smoothed HTTP/TCP throughput estimation to choose the video bitrate. This method compares the segment download time with the segment playout time to increase or decrease the video bitrate. Although this approach can quickly reach the optimum level, it also causes the quality to drop rapidly when the network experiences congestion. Furthermore, this algorithm may fail to provide a fair quality distribution among the different players when they share a common bottleneck [24]. In order to address this challenge, Jiang et al. [72] proposed FESTIVE, which identified the most common issues when multiple players share a common bottleneck: namely, fairness, efficiency, and stability. The authors adopted a set of techniques that systematically achieve video quality in terms of stability, fairness, and efficiency. To do so, a randomised chunk scheduling was first applied to avoid segment overlap between different players. Moreover, with this

approach, the clients slowly switch up the requested bitrate when the actual bitrate of the downloaded segment increases. Furthermore, the bandwidth estimator uses the harmonic mean for the estimated throughput of the last 20 segments. Despite FESTIVE improving the fairness by 40%, stability by 50% and efficiency by almost 10%, it does not support bandwidth fairness among heterogeneous DASH clients who share a common access network.

Another client-based heuristic is CS2P by Sun et al. [144]. This approach is a prediction-based adaptation algorithm that employs throughput prediction for improving end-user video quality. The CS2P model is built offline in a node located in the streaming provider network and deployed online to adapt the rate of the requested video, using a Hidden Markov chain based on the data-driven model to predict the initial throughput and mid-streaming throughput. Experimental results demonstrated that CS2P outperforms other existing approaches by up to 50% in terms of the initial and the midstream throughput prediction. A similar approach was presented in CFA [69], in which the authors cluster the session based on the application layer QoE features (i.e. ISP, geographical region, etc.) rather than throughput prediction accuracy. However, CFA shows that the relation between the application layer QoE features and the predicted throughput is complex.

Similarly, Miller et al. [96] developed a HAS-based live streaming adaptation referred to as LOLYPOP (LOw-LatencY Prediction-based adaPtation). LOLYPOP utilises different time-series prediction schemes to keep the streaming latency within a range of 5 seconds during the rate adaptation. In more detail, LOLYPOP computes the probability of exceeding the playback deadline for the different representations and selects the one with the highest bitrate within the timeframe. In the same context, Abdul et al. [38] presented ABR for Chunked Transfer-Encoding (ACTE), an accurate adaptation scheme for live video streaming. In order to provide a precise bandwidth measurement, ACTE takes the idle time between the consequence chunks into consideration. It therefore computes the throughput based on the consequent chunks, while neglecting any other chunks that arrive after an idle time. Experimental results showed that ACTE achieves a good bandwidth measurement accuracy of 96%, which leads to an 65% reduction in the number of stalls and a 49% increase in QoE.

Buffer-based Approaches

Some studies [24], [61] demonstrated that throughput-based approaches are suboptimal for adapting the requested video when multiple players share a common access network. The inefficiency of throughput-based algorithms worsens in the presence of cache servers. The buffer-based approach was therefore proposed as an alternative approach for estimating the available resources. In [64], Hhuang et al. proposed a buffer-based-approach (BBA). BBA depends on the level of the buffer to make the decision about the requested quality. In more detail, the authors defined two thresholds, Bmin and Bmax, in addition to three levels of the buffer occupancy B: reservoir ($B < Bmin$), upper reservoir ($B > Bmax$) and cushion ($Bmin < B < Bmax$). In the proposed approach, the client requests the lowest or the highest quality if the buffer occupancy is in the reservoir or upper reservoir range.

Furthermore, the authors tested the BBA approach in a field-test environment using Netflix clients. The evaluation results show that BBA reduces the video stalling ratio by 20% compared with the standard adaptive algorithm used by Netflix. Another purely buffer-based approach is the Buffer Occupancy-based Lyapunov Algorithm (BOLA) [127]. In this approach, the authors formulated the ABR as a utility maximisation problem including two KPI metrics: namely, the average video bitrate and the stalling duration. The increasing of the video bit rate increases utility, while rebuffing decreases it. Lyapunov optimisation has been employed within this heuristic to achieve online control and ensure that the achieved utility is within the optimal offline solution. BOLA has been empirically evaluated on a set of test vectors, and it has been found that it is able to achieve 84%-95% of the optimal offline utility.

In the same context, Beben et al. introduced ABMA+ (Adaptation and Buffer Management Algorithm) in [34], which is a buffer-based adaption algorithm that uses the stalling probability to adapt the video bitrate. ABMA+ utilises a previously computed buffer map for finding a bitrate level that offers an adequate buffer level for the end client. Nevertheless, using a pre-computed buffer map minimises the deployment cost and provides the algorithm more scalability to be deployed with different terminals. Experimental results showed that ABMA+ can efficiently adapt the requested video and reduce the switching frequency. Another work was

presented by Yadav et al. in [141], in which the authors employ the queuing theory, particularly the M/D/1/K queue model, to deploy a buffer-based bitrate adaptation algorithm, denoted as QUETRA. The proposed model estimates the level of the buffer occupancy given the current bitrate, calculated throughput, and the current buffer level.

Nevertheless, QUETRA [141] is similar to ABMA+ [34], as it also requires a pre-computed buffer occupancy in implementation. The authors investigated the performance of QUETRA under different network conditions and variant combinations of buffer/segment size. Experimental results revealed the efficiency of QUETRA for providing a stable video delivery despite its simplicity.

Hybrid-based Approaches

As presented in the two previous subsections, end clients can adapt the requested quality based on either throughput or buffer occupancy. However, relying on a single metric may result in sub-optimal selection. Therefore, other works have combined the two metrics for adapting the requested bitrate. Furthermore, different optimisation approaches, such as control theory, dynamic programming, etc., have been employed to optimise the deployment of this approach. One of the early works presented in the literature under this category is Probe ANd Adapt (PANDA) [83]. PANDA extends throughput estimation to be supported with buffer occupancy. It comprises two components, namely throughput estimator and segment scheduler. It also mimics the behaviour of the congestion control at the application layer. Accordingly, based on the estimated throughput, the algorithm slowly increments the quality and decrements it aggressively. It then schedules the next request segments according to the buffer level. The evaluation results demonstrated that PANDA can reduce instability by over 75% when compared with other conventional algorithms. However, there is no explicit rule regarding the implementation of the algorithm.
Further, Yin et al. [145] developed a stochastic optimal control problem to maximise the perceived QoE at the end client. For this reason, the authors proposed a model predictive control-based algorithm (MPC) that maximises end-user QoE by solving the optimisation problem. The proposed algorithm consists of three main components: namely, 'predict', 'optimise', and 'apply'. MPC adapts the quality by

predicting the throughput for the short term. It then finds the optimal bitrate that maximises the QoE of the end-client. Nevertheless, as the point prediction could be inaccurate, the authors proposed RubostMPC, which uses the range prediction instead. For efficiency purposes, an offline computation is conducted for the optimisation problem to create the look-up table that can be used online within the FastMPC version. The experimental evaluation confirms the advantages of MPC compared with the existing algorithms.

SARA [71], or the Segment-Aware Rate Adaptation algorithm, is another example of the mixed-based adaptation approach. It aims to mitigate the misestimation of throughput that could occur due to the variable bit rate (VBR) nature of video data, in which the segmented download time is related to the segment size. To deal with this, authors in [71] utilise the buffer level, throughput measurements, and segment sizes to estimate the optimum bitrate for the next requested segments. To do so, SARA included the size of each segment within the MPD file. Nevertheless, the Weighted Harmonic Mean (WHM) is used together with segment size to provide an accurate throughput estimation. The buffer level is also considered in SARA for adapting the required bitrate. Initial evaluation of SARA indicated the efficiency of including the segment size for providing high throughput estimation.

Recent work in [110] presented a holistic study designed to understand the video encoding characteristics and design of an ABR algorithm that optimises the perceived quality at the end user. In this work, the authors first investigate the relation between chunk quality, scene complexity, and chunk size. As a result, they found that larger segment sizes lead to lower quality, regardless of bitrate level. Scene complexity holds the same property of segment size. In this context, the design of the ABR algorithms should consider the scene complexity of the segment itself. Therefore, the authors use the relative chunk size to distinguish the scene complexity of the different segments. Furthermore, the authors identify the three fundamental design principles (i.e. non-myopic, differential treatment, and proactive) for the ABR algorithm. The first principle (i.e. non-myopic) deals with the bitrate of the next few chunks, while the second principle identifies the complexity of the different chunks. The final principle deals proactively with the large bitrate variability though the target buffer level. To this end, the authors designed CAVA

(Control-theoretic Adaption for VBR-based ABR streaming), a practical ABR rate adaptation algorithm based on control theory. CAVA employs the Proportional-Integral-Derivative (PID) control concepts and deploys the three main principles of ABR design in the implementation phase. CAVA also leads to substantially lower rebuffering (up to 95%) and quality variation (up to 48%).

A recent game theory-based ABR (GTA) was presented by Bentaleb et al. [35]. GTA leverages game theory capabilities [103] and formulates the ABR decisions problem as a bargaining and consensus problem. The proposed algorithm achieves good performance and outperforms its competitors by 38.5% in terms of QoE and 62% in terms of quality stability. Another approach, named Oboe, was presented by Zahaib et al. [26], in which the heuristic-specific parameters can be tuned in real time to meet the current network conditions. Oboe is based on a pre-computed configuration map (ConfMap) to find the best configurations for a given ABR algorithm. The experimental results showed that Oboe significantly improves on client-based adaptation approaches like BOLA [127] and FastMPC [145].

Artificial Intelligence-based Approaches

Authors in [50] adapt the Reinforcement Learning (RL) approach to optimise the adaptation decision at the client side. The proposed adaptation engine performs online learning to maximise the perceived video quality for end-users. In order to solve the optimisation problem, the Markov Decision Process (MDP) has been cast to select the optimal representation of the bitrate level, while simultaneously minimising the occurrence of video stalling and quality switching events. Similar work has been presented by Zhou *et al.* [147], in which the authors propose a Markov decision-based strategy for adapting the requested video. The proposed scheme considers all the QoE-related metrics (i.e. video stalling, quality level, quality switching, end initial delay) within the adaptation process. Furthermore, in order to minimise the optimisation complexity, the authors proposed a sub-optimal greedy algorithm that makes the proposed approach suitable for real-time adaptation. Pensieve [94] is another RL-based video adaptation framework, which learns from previous observations collected from different video players. To do so, Pensieve trains the neural network model to select the optimal bitrate level that matches the network condi-

tions. Experimental results reveal that Pensieve outperforms other benchmarking schemes in terms of achieved QoE by 12%-20%. Gadaleta *et.al* [57] formulate the DASH video streaming problem within an optimisation framework that integrates both the deep learning and reinforcement learning techniques. The proposed framework considers the different QoE related metrics (i.e. video stalling, video quality, etc.) within the QoE optimisation.

The work in [126] proposes a Fuzzy Logic Controller (FLC) that adapts the requested quality at the client side. The proposed controller takes both the estimated throughput and the buffer level as inputs to solve the optimisation problem and schedule the requests to download the video segments in a way that mitigates the ON-OFF pattern. One of the first works to employ the concept of RL within the quality adaptation at the client side was introduced by Jeroen*et.al*. In this approach, the agent learns the best action that maximises the numerical reward based on the given environmental state. The proposed approach reduces the average buffer filling by 8.3% relative to comparison approaches. HASBRAIN [124] is a novel methodology to develop the design of the machine learning-based video adaption framework. In this work, an existing optimisation formulation for finding the optimal decision has been used to identify the best adaptation path. Different models have also been investigated for the training of the adaptation path. The experimental results have demonstrated how HASBRAIN enhances the adaption process even within a challenging mobile environment.

Huang *et.al* [63] present QARC (video Quality Awareness Rate Control), which is a deep reinforcement learning scheme for adapting the video bitrate level. QARC aims to provide a higher video quality and a lower transmission latency through learning from past network conditions and previously downloaded video frames. Huang *et.al* [62] introduce a generative adversarial network (GAN)-based method for adapting the bitrate level, named Tiyuntsong. Tiyuntsong deploys two agents that compete with each other in order to learn the best strategy. Furthermore, to accelerate the training phase, the reward function has been redefined by focusing on the logistic rules rather than the QoE metrics to estimate the perceived QoE. Francis *et.al* [142] recently presented Fugu, a continual learning-based strategy for adapting the video bitrate level selection. Fugu continuously retrains its neural network model

each day based on the observations captured from the last week. Furthermore, instead of estimating the throughput level, it predicts the time required to download a segment of a certain length. Nevertheless, it includes the underlying network statistics within the adaptation process. Evaluation results show that Fugo achieves a lower stalling time and higher video quality compared with other benchmarking algorithms.

3.2.2 Delivery-based Rate Adaptation

Despite the success of the client-based rate adaptation approach for HAS video streaming, as represented by its decentralised and pulling nature, pure client-based adaption may be insufficient for providing fair distributions between users. This is due to the fact that different users have different requirements; for example, users with different device screen sizes should be treated in different ways. The limitations of the client-based rate adaptation approaches prompted researchers to develop various solutions, which aim to provide fair quality distribution and efficient resource utilisation. The following subsections revise the existing research works that have been deployed on different levels, namely server and network levels.

Bitrate Guidance and Resource Allocation

In this category, each client is assisted by an external agent to find the optimal bitrate that fits bot the end-user requirements and the available bandwidth. Deploying such an approach requires exchanging information between the end-users and the quality optimisation agent that can be located within the network. Server And Network-assisted DASH (SAND) [21] was published as an extension for MPEG-DASH to provide the communication channel between DASH clients and the quality optimisation agent. Fig 3.1 illustrates the three communication patterns between DASH clients and a SAND-enabled HTTP server (DANE), namely the Parameters Enhancing Delivery (PED), the Parameters Enhancing Reception (PER) and the Metrics and Status messages. One of the early studies within this category is [107], in which the authors presented a rate adaptation algorithm designed to provide fair bandwidth allocation within a multi-client setting. To achieve this, they deployed an in-network system of coordination proxies to allocate the available network resources

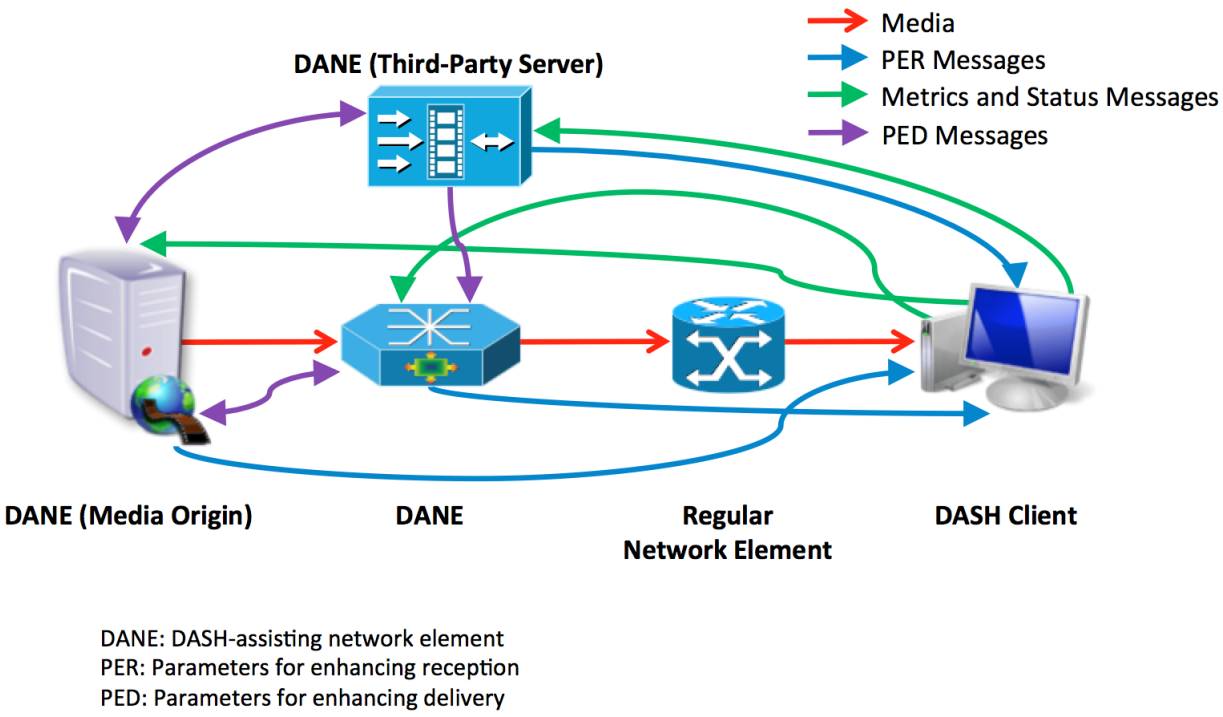


Figure 3.1

SAND Reference Architecture [131]

among DASH players. The in-network proxies periodically monitor the available bandwidth and divide it between the existing clients. As a result, a fair signal is sent to each client to perform the bitrate adaptation. Evaluated bandwidth scenarios demonstrated that the proposed approach improves the quality and the fairness by 20% and 80% respectively compared with the existing reference algorithms.

Kleinrouweler et al. [77] presented an SDN-based DASH-aware networking architecture. In their framework, an external agent presented by the SDN controller provides each client with an optimal maximum bitrate level. The network-based agent guides DASH players with the optimal bitrate by implementing two mechanisms: explicit adaptation assistance and dynamic quality of service. The explicit adaptation assistance is deployed by sending the target bitrate to each client, while later adaptation assistance is implemented through dynamically allocating network resources to each client. The performance evaluation of this approach within a WiFi scenario showed that the received video bitrate could be doubled, while video quality switching is also greatly reduced. Another work is investigated by Cofano et al. [51], in which the authors presented several different network-based schemes for boosting

the performance of the adaptive video streaming services. These schemes are aimed to provide QoE-level fairness among the different players by facilitating cooperation between the network components and DASH players. In order to implement the proposed solution, two strategies (i.e. video bitrate guidance and resource allocation) are implemented. Within the guidance approach, the network component guides each client with the maximum bitrate level, while the network resources are allocated fairly in the second strategy. The perceived QoE was evaluated on video quality, switching frequency, and QoE fairness.

An extended approach was presented by Bentaleb et al. in [36] and denoted as SDNDASH, in which the SDN was leveraged to act as the quality adaptation engine for DASH-player. The external quality engine maximises per-client QoE by dynamically allocating network resources and guiding each client to attain the optimal bitrate level of the next chunk that needs to be requested. Experimental results demonstrate that SDNDASH enhances the received quality by up to 30% compared with other existing approaches. However, this approach also harms the nature of DASH, as the central component is responsible for adapting the quality for each player. Furthermore, it may lead to a single point of failure, as the entire adaptation decision is based on the central component.

Bhat *et.al* [39] propose an SDN-based Adaptive Bit Rate (SABR) architecture, which uses SDN and time series forecasting to assist HAS players with bitrate selection. SABR periodically monitors the network paths between the access network and the CDN caches and provides DASH players with network information, as well as the available bitrate levels of the requested video on the cache servers. Another similar work was presented by Liotou *et.al* [84]; here, the authors proposed the design of QoE-based SDN architecture that offers cooperation between a mobile network operator and video content provider in the context of HAS. According to the proposed design, the network operator reports the network status information to the service providers so that the latter can guide HAS players to proactively cache the requested segments. The proposed architecture utilises the Self-similar Least-Action Walk (SLAW) model to predict user mobility.

Authors in [78] present a predictive Channel Quality-based Buffering Strategy (CQBS), which allows the video buffer to grow when the LTE channel has good

quality and shrinks this buffer if the quality of the LTE channel is low. To obtain the optimal strategy for managing the growth of the buffer, the authors model the behaviour of 377 real-world LTE channel quality traces as a Markov chain. Experimental results show that CQBS improves the stability of the perceived video quality. However, the authors did not consider the fairness between different video players or how available resources can be allocated among different players.

Another work presented by Samain et al. [118] introduces an in-network component that guides the client for the bitrate selection and mitigates the negative impact of deploying video cache servers on client-based bitrate adaption. The authors began by evaluating the impact of deploying cache servers on the decision of the ABR algorithms. It has been noted that using the video cache servers may lead to bandwidth misestimation at the client side, which could result in QoE degradation. To handle the issue under investigation, the authors employed the SAND mechanism to guide the clients to the right bitrate level.

Ozcelik et al. presented a chunk size-aware SDN-based architecture denoted as CSASD [105], which provides DASH players with the bitrate guidance they need to deliver fair QoE for the DASH players while avoiding network underutilisation. To achieve these aims, each player then shares the device characteristics, along with the content details of the requested video, with the SDN-agent, which in turn finds the optimal maximum bitrate. Furthermore, for scalability purposes, CSASD provides the bitrate guidance only at the application level without implementing any resource allocations between background and DASH flows. Experimental results reveal that CSASDN outperforms the purely client-based approaches, increasing the average video bitrate by more than 90% and the decreasing video quality oscillations by more than 84%.

In-network Optimisation

Traffic Re-routing On the other hand, a large body of research focuses on the concept of traffic engineering to optimise the delivery of adaptive video streaming. OpenQoS [56] is one of the earliest traffic engineering approaches for optimising video content delivery. The control plane offers a prioritisation framework that provides a dynamic QoS strategy based on the encoding level of the requested video

segment. OpenQoS considers the enhancement layer of the requested video in terms of best-effort flows, while the based layer is treated as prioritised flows. Another work was presented by Cihat *et.al* [45]; here, the authors present a segment-based routing scheme that utilises the power of SDN when making forwarding decisions. In order to find the optimal path between the source and the destination, the authors consider a set of the context parameters, including the video bitrate level, path throughput, and segment length.

Dobrijevic *et al* [55] adopt ant colony optimisation (ACO) to solve the path selection problem within an SDN-based environment. Furthermore, the QoE estimation model is considered within the presented solution, which in turn provides an intuition about the QoE level based on the network metrics (i.e. packet loss, jitter, etc.). Another work is presented by Bouten *et.al* [42], in which the authors propose an optimisation scheme that enables the client to select the optimal server to stream from. The selection scheme utilises probability-based search strategies to identify the optimal video server. Moreover, in order to avoid video stalling, the optimisation scheme considers the current level of the video buffer within the selection decision. Similarly, Al-Jawad *et.al* [29] introduced LearnQoS, which is policy-based network management (PBNM) that utilises Reinforcement Learning (RL) to provide the optimal decision that best utilises network resources and provides better services for the end client. The authors in [47] formulate the routing problem as an optimisation problem that maximises the perceived QoE rather than the QoS. Furthermore, a QoS-QoE model has also been developed that incorporates the QoE-related metrics to estimate the end-user QoE. For scalability purposes, the authors use the Lagrange decomposition and sub-gradient optimisation. The k-shortest path algorithm is then used to calculate the shortest path for each sub-problem.

Huang *et.al* [65] leverage Deep Reinforcement Learning (DRL) to present a traffic control framework for maximising multimedia services within SDN environments. In order to achieve optimal results, QoE is used as the main indicator for the control strategy. Furthermore, to accelerate the convergence speed of the optimisation problem, Deep Deterministic Policy Gradient (DDPG) is used to solve this problem. Khalid *et.al* [73] present an SDN-based framework, Device-Aware Network-assisted Optimal Streaming service (DANOS), that enables interaction between the service

provider and the network operator. These authors further formulate a QoE maximisation problem that considers device capabilities, subscription levels, and network constraints. Moreover, the authors in [119] introduce an SDN-based architecture that offers service differentiation among the different DASH players. Mixed Integer Programming (MIP) is considered to solve the optimisation problem and find the optimal path between server and the client. Moreover, in order to reduce the computational complexity, a decomposition-based heuristic algorithm is used.

Stream Prioritisation Prioritisation is understood to be an efficient method of optimising the QoE level of the end-user, particularly in the case of video freezes. Pietrangelo et al. [108] proposed a centralised SDN-based framework that prioritises the delivery of certain video segments in order to prevent stalling. Their framework is based purely on the OpenFlow statistic on the network nodes to estimate the buffer level of the end-user, and also requires no feedback from the end-user, which in turn increases the scalability and the range of the served users.

The information collected from the network nodes is then fed to the control level, which is a Machine Learning (ML) engine. The random under-sampling boosting algorithm and fuzzy logic are adopted within the ML engine to estimate the video buffer level at the client-side; therefore, it can prioritise the video traffic before video stalling occurs. This framework provides a scalable mechanism, which enables it to reduce video stalling events with no interaction with the video players.

A similar approach is presented by Zinner et al., [149], which also uses the level of the video buffer to trigger the prioritisation process. The main difference with the previous approach is that this framework facilitates collaboration between the end-users and the network controller via sharing the buffer level. This approach is therefore less scalable than the [108] framework, which is based only on the network nodes to estimate the buffer level. In this framework, different queues with different prioritisation levels are installed on the forwarding devices. Video traffic is therefore forwarded through a dedicated queue when the corresponding buffer level runs below the predefined threshold. The priority queuing approach is also compared in this work with a weighted fair queuing approach; it is found that the latter performs better, as it offers a better mechanism for allocating the available network resources. The work in [31] [30] presented an SDN-based resource allocation

scheme for which, user traffic is profiled based on application usage trends.

Proxy-based Similar work has been presented by Georgopoulos et al., [58], in which the authors introduce a QoE Fairness Framework (QFF) that achieves QoE-level fairness among the different DASH players. For this purpose, the authors rely on the OpenFlow features to implement the desired QoE policy at the network level. Two matters are considered in this work: first, the nonlinearity between the perceptual quality and the video bitrate level; second, that equal allocation of the available network resources among the different DASH players may result in unfair QoE distribution among the different players. Nevertheless, the proposed framework consists of different components that interact with each other to achieve the desired purposes. The utility and optimisation functions are the primary components of QFF. The former function maps the video bitrate level to the perceived video quality; on the other hand, the optimisation function solves the QoE maximisation problem and ensures QoE-level fairness among the different DASH players.

Furthermore, Mansy et al. [93] show that QoS-level fairness may not always result in QoE-level fairness. They accordingly present a new QoE-based metric that considers both screen resolution and the distance between the end-user and the screen. Moreover, the concept of max-min fairness is applied to solve the QoE distribution problem.

Mok et al., [99] presented QoE-aware DASH (QDASH), a measurement proxy located between the end-user and the video streaming server. In QDASH, the proxy server is adopted to measure the available bandwidth, round-trip time and loss rate. According to this measurement, QDASH can guide the end-user towards the bitrate most suited to the network conditions. The authors also proposed an intermediate quality level as a bridge between the original level and the estimated (lower) quality lever, which can improve the received QoE. Nevertheless, the authors in [74] presented a predictive mechanism that manipulates network packets differently according to their video frame type. Authors in [76] investigated the impact of the well-known scheduling approaches on the delivery of video traffic. To do the authors compared the content-aware and content unaware strategies in terms of their impacts on the network metrics and the user-centric metrics. Experimental results showed that the content-aware strategy could improve user satisfaction.

Transport Level Optimisation in SDN-assisted Networks

Wu *et. al* [140] present a quAlity-Driven MultIpath TCP (ADMIT) framework, which aims to improve the perceived video quality at the end user through enhancing the MPTCP performance. The authors investigate some QoE issues that could potentially emerge when the MPTCP is used in heterogeneous network terminals. In order to mitigate the distortion of the End-to-End video streaming, this framework incorporates the quality-driven Forward Error Correction (FEC) coding and rate allocation. Similarly, Corbillon *et. al* [52] present a solution for eliminating the concern with using the MPTCP regarding the transportation of video content. Head-of-line blocking, delay, and throughput instability are among the main drawbacks of the transport protocol. To enhance the performance of the MPTCP, the authors present a cross-layer scheduler that enables the interaction between the application and the transport layers without significant overhead. Another work presented by James *et. al* [68] aims to answer the question of how MPTCP is beneficial to mobile video streaming. Evaluation results indicate that using MPTCP is not always necessary, especially when the main primary path has sufficient available bandwidth. The use of MPTCP can also be harmful to the perceived QoE when the secondary MPTCP path exhibits unstable network conditions. Another issue with MPTCP is that it lacks support for prioritising different network interfaces. To cope with this issue, Ham *et.al* [59] propose a framework called MP-DASH that considers user preferences within the data scheduling. Experimental results show that MP-DASH can reduce cellular usage and radio energy consumption by 99% and 85% respectively. Nam *et. al* [67] used SDN to improve the performance of the MPTCP and enhance the delivery of the video content. In more detail, the authors leverage the SDN controller to provide a dynamic forwarding strategy for video traffic.

Chapter 4

A Max-Min Bounded Bitrate Guidance for SDN Enabled Adaptive Video Streaming

4.1 Introduction

The increase in video traffic and users' demand for higher quality video has motivated service providers (SPs) and mobile operators to find novel bandwidth management solutions and better resource utilisation methods for their existing network infrastructure in order to optimise the QoE experienced by the end users. However, achieving a good QoE, as explained in section 1.2, is a challenging task due to the wide range of client patterns (i.e. device and video characteristics) and the variability of the network conditions [23].

As presented in Chapter 3, purely client-based HAS solutions face challenges related to quality fluctuations when competing for a shared link [24]. The first challenge relates to the mismatched ON/OFF behaviour of the different players, which can lead to inaccurate throughput prediction. Furthermore, other QoE related metrics, such as the impact of the end-user device and the user context, are not considered within the ABR algorithms [24], which might result in unfair quality distribution among different users.

Nevertheless, some of the solutions presented in Chapter 3 that aim to overcome the issues with client-based HAS solutions were based on the assistance of a network-

based component, in which the network and video providers exchange information and cooperate. Emerging technologies, such as the server and network-assisted delivery (SAND) architecture [130], offer standard signalling for network-to-client and network-to-network communication of quality-relevant information. This promising architecture enables the network elements to apply appropriate policies that match network condition with user requirements. SDN [79] as discussed in Chapter 2, is an emerging technology used to deploy such architecture, and can provide centralised control for efficient and flexible network management. However, moving the adaptation decision to an external entity harms the underlying principle of DASH and could thus result in many issues.

In order to overcome the above issues, this Chapter provides threefold contributions towards improving the perceived QoE for video streaming [28]. First, the chapter introduces a novel guidance mechanism, called a Bounded Bitrate Guidance for DASH (BBGDASH). Unlike the fully centralised network guidance [36], the proposed guidance provides guidance to the client without the need to move the whole control logic to an additional entity nor relying purely on the client-side decision. Second, this work provides an implementation of a network-assisted video streaming framework that leverages the SDN functionality to deploy BBGDASH. Third, a use-case based evaluation has been conducted to assess the feasibility and the potential of the proposed approach.

The remainder of this chapter is structured as follows. Section 4.2 introduces the SDN-based architecture of BBGDASH. Section 4.3 provides the system modelling and presents the guidance algorithm used to compute the boundary bitrate levels for each DASH player. Section 4.4 details the experimental setup. The performance evaluations of the proposed approach are then outlined in Section 4.5. Finally, Section 4.5.1 provides a summary of this chapter.

4.2 The Proposed System Architecture for Video Streaming using SDN

The BBGDASH architecture leverages the features and capabilities of SDN in order to dynamically manage the available resources among users in a way that provides

them with a fair QoE allocation. As shown in Figure 4.1, the proposed solution consists of three layers: the application layer, the control layer, and the infrastructure layer. Several functional entities and communication interfaces are distributed throughout these three layers. The design of BBGDASH's architecture will be discussed in more detail below.

4.2.1 Application Layer

This layer is composed of three entities: namely, the DASH server, the DASH player and the SDN-based DASH agent.

DASH Server

This entity is represented by a standard HTTP server, which stores and manages video content. Each video is encoded with different bitrate levels and divided into small segments with a typical duration of 2–12 seconds. Furthermore, each video is associated with a Media Presentation Description (MPD) file that embeds video features, including the available bitrate levels and resolutions, segment size, etc. Nevertheless, objective quality measurement has also been provided for each video based on the Structural Similarity Index (SSIM) [139] to be used as a reference metric for measuring the perceived quality.

DASH Player

In order to exchange the information between the DASH players and the SDN-based agent, the dash.js reference player requires modest modifications so that it can apply the received bitrate guidance and send the required stats to the network side. Accordingly, on the DASH player side, once it has received the MPD file, the quality-related parameters (along with the detected screen resolution, which is captured on the application layer) are sent to the DASH manager entity within the SDN-based agent to define the bitrate levels for each DASH player. Moreover, each DASH player utilises the received signal (i.e. bitrate levels) to adapt its bitrate locally within the suggested bitrate levels via the SDN-based agent. Nevertheless, other QoE indicator metrics – such as the average bitrate, initial buffering time, number of stalling, stalling duration, and the number and amplitude of quality

switching – are collected at the DASH player side and sent to the SDN-based agent to be used for monitoring and the evaluation purposes.

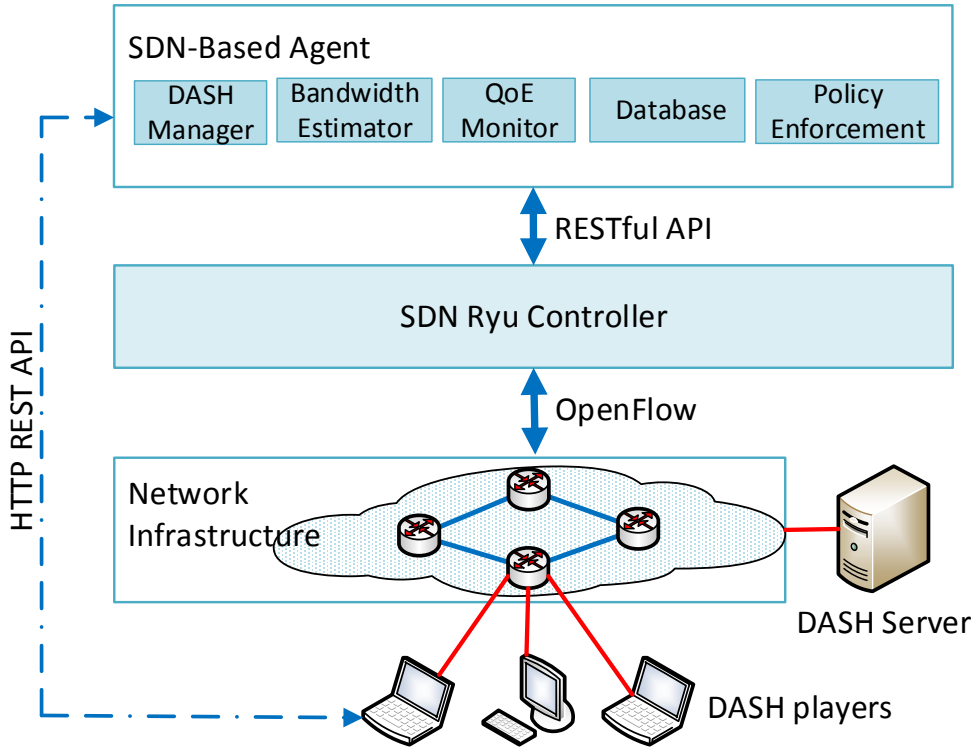


Figure 4.1
BBGDASH Architecture

SDN-Based DASH Agent

This agent is implemented on top of the control plane, which consists of five modules: namely, the DASH manager, bandwidth estimation, QoE estimation, database, and policy enforcer. In the following, each is described in detail.

DASH manager:

This module interacts with other components within the SDN-based agent in order to obtain the required information about the active DASH players and the underlying network conditions. Furthermore, the collected information is used as an input for the proposed algorithm in section 4.3, which in turn finds the optimal bitrate levels for each DASH player and allocates the network bandwidth in a way

that utilises the available bandwidth and achieves fair QoE distributions among the DASH players.

Bandwidth Monitor:

Accurate bandwidth estimation is a critical prerequisite for computing the optimal boundary bitrate levels that fairly maximise the perceived quality within DASH players and achieve an efficient utilisation for the available network resources. To achieve this, the module frequently requests the control plane, which in turn sends the proper rules that enable it to acquire the network stats of the network components within the data plane. The weight exponential moving average is then applied to the collected network measurements to obtain a smoothed bandwidth estimation.

QoE Monitor:

The primary purpose of this module is to estimate the perceived QoE at the end DASH player. Therefore, it considers the main QoE metrics (i.e. average bitrate level, initial delay, stalling and video bitrate switching) and applies a proper QoE model that maps the collected QoE into an equivalent index of MOS.

Database Module:

This module acts as an interface between the database engine and the other modules within the SDN-based agent. Two types of parameters are considered within the design of the database scheme: namely, the operation-based and evaluation-based parameters. The operation-based parameters include all metrics that are used as inputs to run the quality optimisation algorithm, including the number of Active DASH players, the available network resources, the experimental parameters, the screen resolutions for each DASH player, and the video features of each requested video, while the evaluation parameters comprise all metrics used to conduct the performance evaluations.

Policy Enforcement:

The main rule underpinning this module is to carry out the outcomes of the quality optimisation algorithm on the network and application levels. The network-level

action is implemented through the SDN controller via network resource slicing, while the application-level actions can be applied by sending the recommended maximum and minimum bitrate levels (i.e. bitrate range) for each DASH player.

4.2.2 Control Layer

This layer acts as the bridge between the application and infrastructure layers. It has been designed to offer a set of QoE-related services for the two layers. It further uses RESTful [148] to communicate with the application layer, while OpenFlow [95] is used to provide communication with the infrastructure layer.

4.2.3 Infrastructure Layer

This layer represents the set of OpenFlow-enabled forwarding devices, such as switches and routers. These devices are responsible for packet forwarding and allocating the resources based on the SDN controller policies. In this thesis, an Open VSwitch (OVS) that acts as the forwarding device of the proposed architecture is employed.

4.3 System Model and Algorithm Description

At each time t, most existing ABR algorithms aim to address the resource provisioning problem for N concurrent players that access a shared bottleneck link with a limited bandwidth of BW. Consider an example in which three DASH players request videos with maximum bitrate levels 3.8, 4.5, and 3.5 Mbps respectively, while the available network bandwidth is 8 Mbps. This requires a potential maximum bandwidth $BW_{max} > BW$ in order to allow each player $p_i \in P$ (P denotes the set of players) to stream with the maximum bitrate level $l_{p_i}^{M_{p_i}} \in L^{v_{p_i}}$ of the requested video $v_{p_i} \in V$, such that:

$$\sum_{i=1}^{N} l_{p_i}^{M_{p_i}} < BW, \forall p_i \in P, i = [1, \ldots, N] \tag{4.1}$$

Here, V is the set of videos stored in the DASH server, while $L^{v_{p_i}}$ is the set of bitrate levels of video v_{p_i} requested by player p_i.

It is assumed that each player $p_i \in P$ has a specific device resolution r_{p_i}, and may request a video $v_{p_i} \in V$ that is encoded in a distinct number of bitrate levels $M_{pi}, i = [1, \ldots, N]$; here, each bitrate level is represented by $l_{p_i}^{j} \in L^{v_{p_i}}, j = [1, \ldots, M_{p_i}]$, so that it can be presented as follows:

$$\frac{\sum_{i=1}^{N} M_{p_i}}{N \times M_{p_i}} \neq 1, \frac{\sum_{i=1}^{N} l_{p_i}^{M_{pi}}}{N \times l_{p_i}^{M_{p_i}}} \neq 1, \forall p_i \in P, i = [1, \ldots, N] \tag{4.2}$$

In this context, enforcing the same bitrate allocation for all users may result in an unfair distribution and inefficiency of both the QoE and network resource allocation. In order to meet the user QoE requirements and achieve a high level of fairness among users requesting different videos, in terms of video type, number of bitrate levels and content resolution, a network-assisted approach has been developed based on the concept of *max-min fairness* [81].

In the proposed approach, the system is modelled as an undirected graph $G = (X, Y)$, where X represents the set of nodes x and Y is the set of links y between the nodes. The set of nodes in the proposed architecture includes several subsets, encompassing DASH players P, the DASH server S, forwarding devices F, and the SDN controller A. Each player $p_i \in P$ has at least one link $y \in Y$ that connects it to the access node $f \in F$. It is further assumed that the network has only one bottleneck in the access network with a total capacity of BW at time t. At any time t, each player p_i requires $BW_{p_i} \in \mathbb{R}+$ to stream a video v_{p_i} with a bitrate level equal to $l_{p_i}^{j} \in L^{v_{p_i}}$. Furthermore, the total bandwidth allocated to all players must not exceed the total capacity BW of the shared link, as in the following:

$$\sum_{i=1}^{N} BW_{p_i} \leq BW, \quad \forall p_i \in P, i = [1, \ldots, N] \tag{4.3}$$

The main objective is to maximise the minimum bitrate among users. The

objective function f is described as follows:

$$f = \begin{cases} \max\left(min\ l_{p_i}^{j} \in L^{v_{p_i}}\right) \\ s.t\ \sum_{i=1}^{N} BW_{p_i} \leq BW, \\ \quad BWp_i \geq 0, l_{p_i}^{j} \in L^{v_{p_i}} \\ \quad \forall p_i \in P, i = [1, \ldots, N], j = [1, \ldots, M_{p_i}] \end{cases} \tag{4.4}$$

To solve the objective function (Eq. 4.4), a dynamic programming-based algorithm (Bitrate Selection Algorithm 1) is used to calculate the optimal boundary range for the maximum and minimum bitrate levels for each player. The proposed algorithm is based on the fact that different players, with their different requirements, may request videos with heterogeneous representation levels, therefore dividing the available bandwidth equally among different users; this results in unfair and inefficient resource allocation, and thus, a poor QoE distribution among users.

The algorithm begins by initialising the input variables, including N, $L^{v_{p_i}}$, BW, Cl, N_{cl_k}, $l_{cl_k}^{opt}, and\ \alpha$, which are defined along with corresponding notations in table 4.1. In line (1), the algorithm computes the scaling factor S_f for each cluster $cl_k \in CL$, since each cluster includes a set of players that have the same screen resolution r_{cl_k}. Based on the computed S_f, the algorithm computes the initial optimal bitrate level $l_{cl_k}^{ini,p_i}$ for each player p_i in cluster cl_k (lines 2–3). Next, for each player p_i within cluster cl_k, the algorithm finds $l_{p_i}^{\sigma}$ as the higher bitrate level less than or equal to $l_{cl_k}^{ini,p_i}$ to be stored in V_1 lines (4–7). Lines (8–15) compute the bandwidth required for each player to stream with the next bitrate $l^{\sigma+1}p_i$. The amount of bandwidth remaining BW^{left} from the initial allocation is calculated in line (16). Lines (17–23) fairly redistribute the remaining bandwidth among players in a way that ensures a fair QoE distribution and efficient resource utilisation. Based on the value of α (i.e. the difference between the minimum and maximum bitrate levels), the algorithm defines the lower bound for the selected bitrates in lines (24–25). The algorithm ends by sending the minimum and maximum bitrate level for each player to the Policy Enforcement within the SDN-based Agent.

Algorithm 1 Bitrate Selection Algorithm

Input: $N, L^{v_{p_i}}, BW, Cl, N_{cl_k}, l_{cl_k}^{opt}, \alpha$

1: CALCULATE $S_f \leftarrow \frac{BW}{\sum_{k=1}^{Z} N_{cl_k} * l_{cl_k}^{opt}}$
2: **for each** *cluster* $cl_k \in Cl$ **do**
3: $\quad l_{Cl_k}^{ini,p_i} \leftarrow l_{cl_k}^{opt} * S_f$
4: **for each** *cluster* $cl_k \in Cl$ **do**
5: $\quad$ **for each** *player* $p_i \in P$ **do**
6: $\quad\quad$ FINDMAX$(l_{p_i}^{\sigma} \in L^{v_{p_i}}) \mid l_{p_i}^{\sigma} \leq l_{Cl_k}^{ini,p_i}$
7: $\quad\quad$ INSERT$(l_{p_i}^{\sigma})$ *into* $V_1[]$
8: $\quad$ **for each** *player* $p_i \in P$ **do**
9: $\quad\quad$ **if** $l_{p_i}^{\sigma} \neq l_{p_i}^{M_{p_i}}$ **then**
10: $\quad\quad\quad$ FIND$(l_{p_i}^{\sigma+1})$
11: $\quad\quad\quad$ $l_{p_i}^{diff} = (l_{p_i}^{\sigma+1}) - (l_{p_i}^{\sigma})$
12: $\quad\quad$ **else**
13: $\quad\quad\quad$ $l_{p_i}^{\sigma+1} = \infty$
14: $\quad\quad$ INSERT$(l_{p_i}^{\sigma+1})$ *into* $V_2[]$
15: $\quad\quad$ INSERT$(l_{p_i}^{diff})$ *into* $V_3[]$
16: $\quad$ CALCULATE$(BW^{left}) = \sum_{i=1}^{N} l_{Cl_k}^{ini,p_i} - l_{p_i}^{\sigma}$
17: $\quad$ SORT$(V1)$
18: $\quad$ **while** $BW^{left} \geq$ MIN(V_3) **do**
19: $\quad\quad$ **for each** $l \in V_1$ **do**
20: $\quad\quad\quad$ **if** $V_3[l] \leq BW^{left}$ **and** $V_2[l] \neq \infty$ **then**
21: $\quad\quad\quad\quad$ $V_1[l] = V_2[l]$
22: $\quad\quad\quad\quad$ $BW^{left} = BW^{left} - V_3[l]$
23: $\quad\quad\quad$ **else** CONTINUE()
24: $\quad$ **for each** $l \in V_1$ **do**
25: $\quad\quad$ $V_2[l] = V_1[l] - \alpha$
26: $\quad$ RETURN(V_1, V_2)

Table 4.1

Notation and Symbols Description.

Symbol	Descriptions
N	Total Number of DASH players
Cl	Set of clusters $cl_k, k = [1, \ldots, Z]$ of specific screen resolution r_{cl_k}
N_{cl_k}	Number of DASH players in cluster cl_k
$l_{cl_k}^{ini,p_i}$	Initial bitrate selection for player p_i in cluster cl_k
$l_{cl_k}^{opt}$	Optimal bitrate for cluster cl_k
S_f	Scaling factor
BW	Total capacity of the bottleneck link at time t
BW_{p_i}	Bandwidth consumed by player p_i
BW^{left}	Remaining bandwidth after the first allocation
$L^{v_{p_i}}$	The complete set of bitrate levels $[l_{p_i}^1 : l_{p_i}^{M_{pi}}]$ of video v_{p_i} requested by player p_i
$l_{p_i}^{diff}$	Difference between the current bitrate level and the next level of the video v_{p_i} requested by player p_i
V_1	Maximum allowed bitrate set for the players
V_2	Minimum allowed bitrate set for the players
V_3	The difference between the current bitrate level and the next bitrate level for the players
α	The difference between max. and min. selected bitrates

4.4 Experimental Setup

The SDN architecture proposed in Section 4.2 was implemented in a testbed environment to evaluate the performance of BBGDASH. In this section, the testbed and methodology used for the experiment are explained.

4.4.1 Evaluation Testbed

The set of experiments were conducted on the proposed SDN-based solution, which was implemented on three virtual machines (VMs) running Linux (Ubuntu V16.04 LTS) as shown in Figure 4.2. Our testbed is divided into three planes: data plane, control plane, and application plane. The data plane is implemented on the Mininet [14] V2.3 network emulator and consists of nine DASH players (U1, U2, ..., U9) and two OpenFlow switches. The control plane is implemented using a Ryu SDN controller [17] and employs the OpenFlow v1.3 protocol as the southbound interface to apply the required QoS configurations and periodically pull network statistics. RESTful API is also used as the northbound interface to provide communication between the Ryu controller and the application plane that hosts dash.js [7]-based players running on Google Chrome. In addition, an Apache server has been attached to the Mininet network to provide video access for the DASH players. Furthermore, in order to provide varying content in the video server, the Big Buck Bunny video has been used; this video has a duration of 600 seconds and is encoded with the H.264 codec using FFmpeg [8] at five different resolutions (240p, 360p, 480p, 720p, and 1080p) with varying bitrate representations, as shown in table 4.2. Subsequently, each bitrate is segmented into sets of six-second chunks using GPAC MP4Box [10]. The bitrate levels were selected based on the DASH dataset [5] in a way that offers a continuous quality increase across the bitrate levels.

On the client side, the dash.js player [7] was extended by building an HTTP communication channel with the database server (MySQL V5.7) to report the KQIs and the content information in a real-time manner. Furthermore, the database server provides access for the SDN applications to inquire about the QoE-related parameters and configure the available network resources based on the acquired information.

4.4.2 Experiment Design

The implemented testbed was used to provide access for nine DASH players that share a common access network for video streaming purposes. This replicates the scenario of a moderately congested residential network with multiple users aiming to stream videos concurrently [16]. In this scenario, the first three DASH players

Table 4.2

Video Representations for Big Buck Bunny

Bitrate	**Resolution**	**PSNR**
128 kbps	320x240	36.5
255 kbps	480x360	37.6
320 kbps	480x360	38.2
500 kbps	854x480	39.1
780 kbps	1280x720	40.2
1000 kbps	1280x720	41
1200 kbps	1280x720	42
1460 kbps	1280x720	43
2000 kbps	1920x1080	43.8
2400 kbps	1920x1080	44.3
2900 kbps	1920x1080	45.0
3300 kbps	1920x1080	46.2
3500 kbps	1920x1080	47.1
3800 kbps	1920x1080	48.2

are requesting a video encoded at eight bitrate levels L1 = {128, 255, 320, 500, 780, 1000, 1200, 1460} Kbps, while the second three players are requesting a video encoded at 13 bitrate levels L2 = {128, 255, 320, 500, 780, 1000, 1200, 1460, 2000, 2400, 2900, 3300, 3800} Kbps; moreover, the remaining players are requesting a video encoded into 11 bitrate levels L3 = {128, 217, 373, 573, 779, 1200, 1460, 2000, 2900, 3500, 3800} Kbps. The bitrate levels were selected based on the DASH dataset [5].

In order to create a shared bottleneck between players, the link between OVS1-OVS2 was shaped to 21 Mbps using the Traffic control (*tc*) tool for bandwidth shaping. The evaluation included a set of four experiments, as discussed below.

***First Experiment*:** The goal of the first experiment was to evaluate the stability of the received video and the efficiency of utilising the available resources when

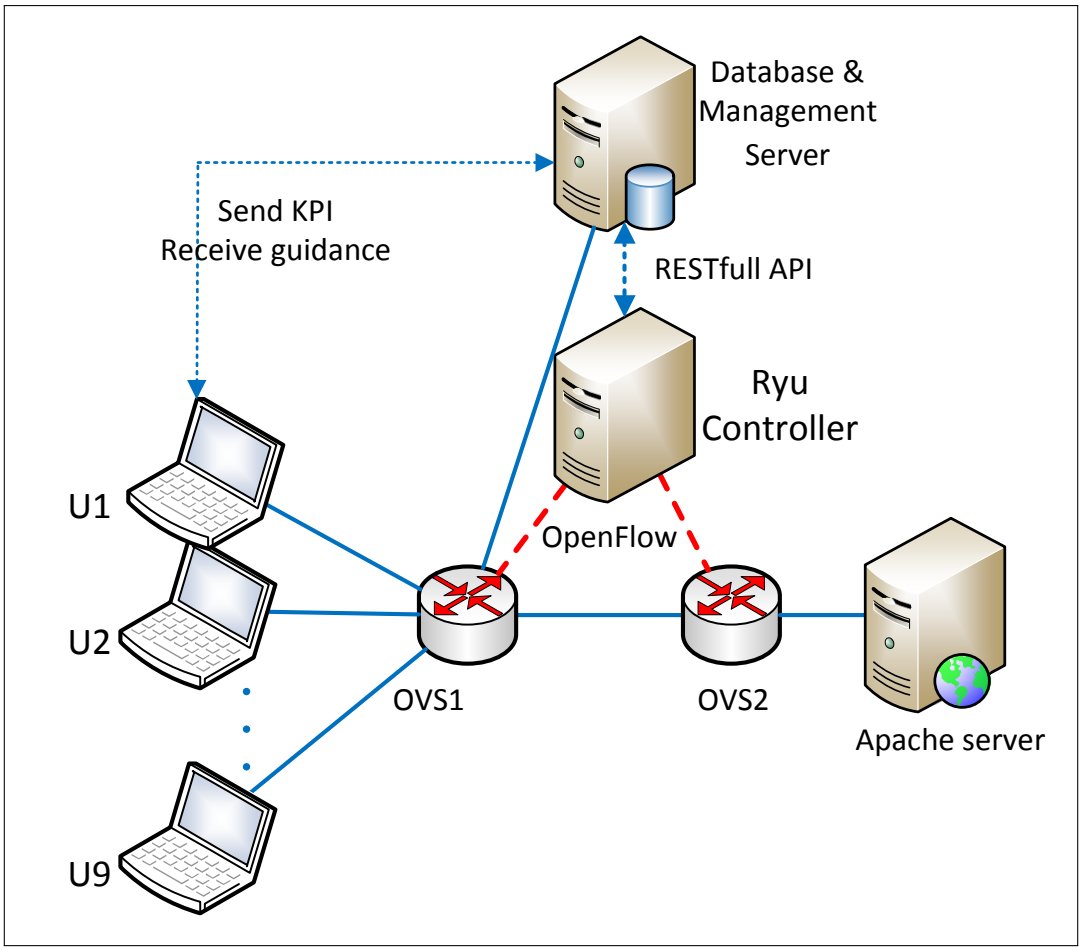


Figure 4.2

Experimental Testbed

nine DASH players stream video without application/network assistance. In this experiment, each player relies only on the local ABR algorithm to adapt the quality of the received video to the network conditions and playback buffer occupancy.

***Second Experiment*:** The second experiment replicates the work of Kleinrouweler et al. [77], which enabled an application/network assistance approach to provide stable video delivery. The purpose of the second experiment is to demonstrate the feasibility of allocating the available resources equally without considering the device or video specifications.

***Third Experiment*:** In the third experiment, the proposed network assistance approach for DASH streaming was applied to provide each player with only the maximum allowed bitrate level, denoted as BBGDASH1/0.

This experiment investigates how the proposed solution can outperform the purely client-based DASH solution and the compared network-based assistance ap-

proach outlined in [77] when different players request videos with varying bitrate representations. This experiment also assesses the efficiency of guiding DASH players with only the maximum allowed bitrate level and without bounding the requested bitrate levels.

***Fourth Experiment*:** The efficiency and the performance of the bitrate guidance approach were benchmarked in the fourth experiment, in which each player receives the maximum and minimum allowed bitrate levels (i.e. BBGDASH1/1) that define the range of the requested bitrate. The aim of this experiment was to investigate whether bounding the interval of bitrate values can have additional benefits when compared to specifying only the upper boundary.

In order to compare the performance of the four approaches, a set of metrics have been adopted [121] [37] [145] [81] to measure the stability of the received video, the efficiency of utilising the available resources, the perceived QoE, and the fairness level among different players. For the purpose of investigating the stability of the received video, the number and amplitude of the bitrate switching [121] were used as performance metrics. The underutilisation index metric, as introduced in [37], was adopted for measuring how efficient the proposed approach is at utilising the available network resources, in which the value closest to zero represents the most efficient utilisation of the available resources. The received QoE was calculated using the model presented in

[145], which is based on the four main QoE metrics (i.e. average bitrate, bitrate switching, video stalling, and initial delay) used to calculate the perceived QoE. To provide a standard measurement for QoE, the results of the QoE model have been normalised by dividing them over the optimal QoE value that can be perceived under the best conditions. Further, Max ratio [81] was used in our evaluation to examine the fairness level among players.

4.5 Experimental Results

This section compares the potential performance of the proposed solution with the other benchmarked approaches. The evaluation begins by comparing the different approaches in terms of the received bitrate levels, followed by evaluating the impact

of the different approaches on video delivery stability. Moreover, the effect of the different approaches on the utilisation of available network resources has also been investigated. Finally, the influence of the benchmarked approaches on the perceived QoE and QoE level fairness is examined.

Selected Video Bitrate

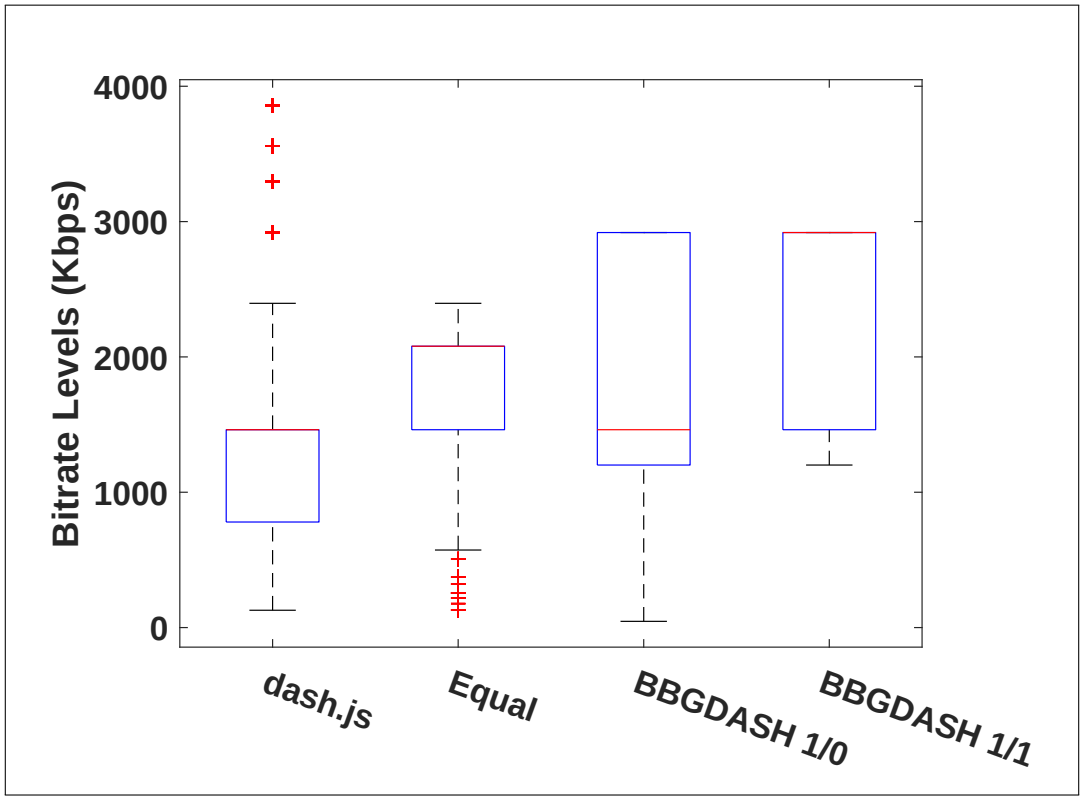


Figure 4.3
Bitrate Selected

Figure 4.3 compares the proposed solution with other benchmarked approaches in terms of the achieved bitrate level. As the figure shows, in the absence of application/network assistance (i.e. purely ABR-based streaming), providing stable video delivery is inefficient when multiple players share a common bottleneck. This inefficiency arises because local ABR algorithms adapt their quality in a greedy way, attempting to stream at the highest video quality possible. This behaviour results in frequently switching to the lowest bitrate level to avoid stalling. The application/network assistance approach of Kleinrouweler et al. [77] is applied to guide the users to identify the maximum allowed bitrate level results while boosting the

bitrate level of the streaming video. Figure 4.3 shows that the median bitrate level of all users increased from 1462 Kbps without assistance to 2079 Kbps under the benchmarked bitrate guidance scheme of [77]. However, since the different players may have different characteristics (i.e. screen resolution), they may request videos with varying bitrate representations; therefore, treating the users equally based on the Kleinrouweler et al. [77] approach could lead to inefficient resource allocation and poor QoE distribution among the different players. Figure 4.3 illustrates the gain of applying the proposed algorithm based on the upper bound guidance (BBGDASH1/0) in terms of maximising the received bitrate.

Although the upper bound guidance approaches seem to improve the perceived quality of the delivered video by guiding each player to find the maximum allowed bitrate level that should be requested, there is no guarantee that the players will actually reach the recommended bitrate. Figure 4.3 reveals that, based on the upper bound guidance approaches, the average bitrate level for all players is less than the bitrate recommended by the agent.

This prompts the conclusion that sending only the maximum allowed bitrate level is not always an efficient way to deliver video with the recommended bitrate level. In order to deal with this issue, the bitrate guidance scheme has been adjusted by sending the minimum and maximum bounds (i.e. BBGDASH 1/1) that should be requested. Figure4.3 also indicates that BBGDASH 1/1 boosts the bitrate level of the received video to ($M = 2900$ kbps), compared to ($M = 1462$ kbps) with the upper bound bitrate guidance.

Number of Video Switching Events and their Amplitude

Regarding the stability of the delivered video, Figures 4.4 and Figure 4.5 show the stability of the received video for the four approaches in terms of the number and amplitude of the bitrate switching respectively. In Figure4.4, the guidance-based approaches outperform the unassisted dash.js in terms of stability. The total switching number dropped from 240 to 130 for equal allocation upper bound-based guidance and to 110 for BBGDASH 1/0, while BBGDASH 1/1 provides more stability by reducing the bitrate switching number to 30. Likewise, the amplitude of bitrate switching has a different effect on the user experience, as a switching event with a

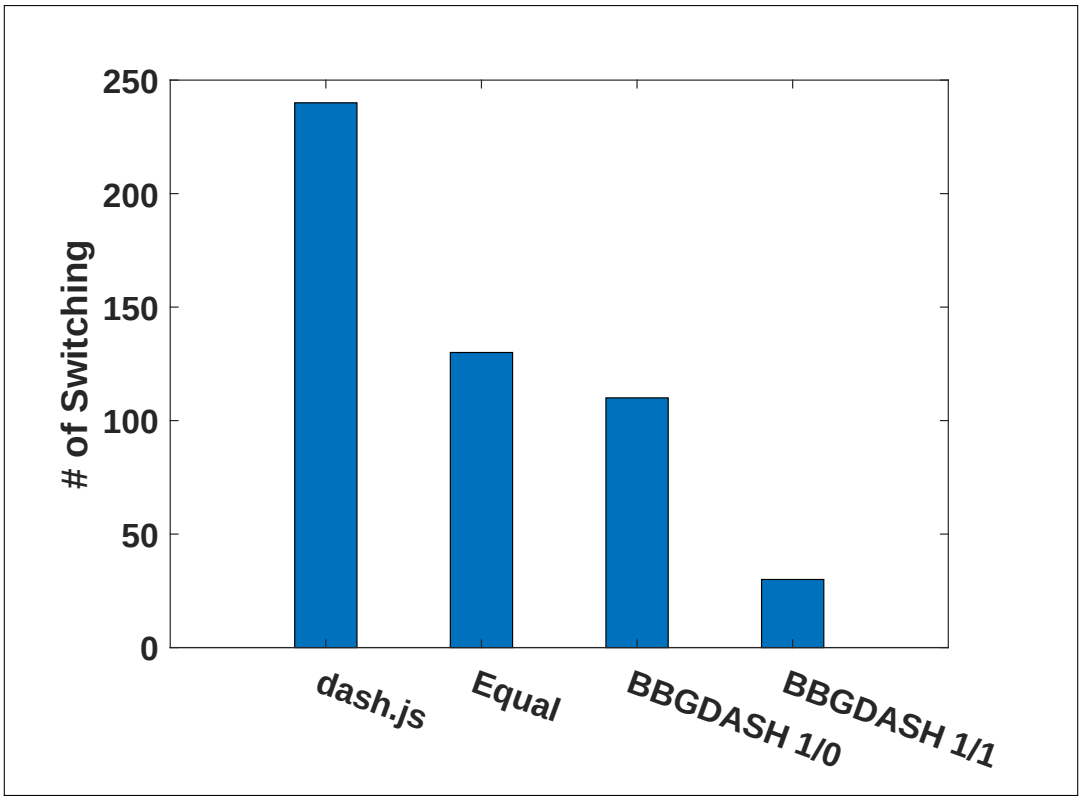


Figure 4.4
Number of Bitrate Switching Events

low amplitude degrades the user's watching experience only negligibly. Figure 4.5 compares the amplitude of bitrate switching for the four considered approaches. It is clear that (BBGDASH 1/1) can reduce the amplitude of the switching, as the player should not switch to a bitrate level below the minimum allowed bitrate level while the buffer level exceeds the minimum threshold. Figure 4.5 shows that the maximum switching amplitude value of BBGDASH 1/1 has a value of 2, compared to 12 for the upper bound guidance and 18 for the unassisted dash.js.

Underutilisation Index

Resource utilisation for the different approaches is measured based on the under-utilisation index, which has been defined in section 4.4. Figure 4.6 depicts the value of the underutilisation index for the examined approaches. It is clear that BBGDASH 1/1 outperforms the other approaches in terms of utilising the available resources, with a low value of the underutilisation index equal to ($M = 0.002$). This can be contrasted with the other approaches, which performed inefficient resource

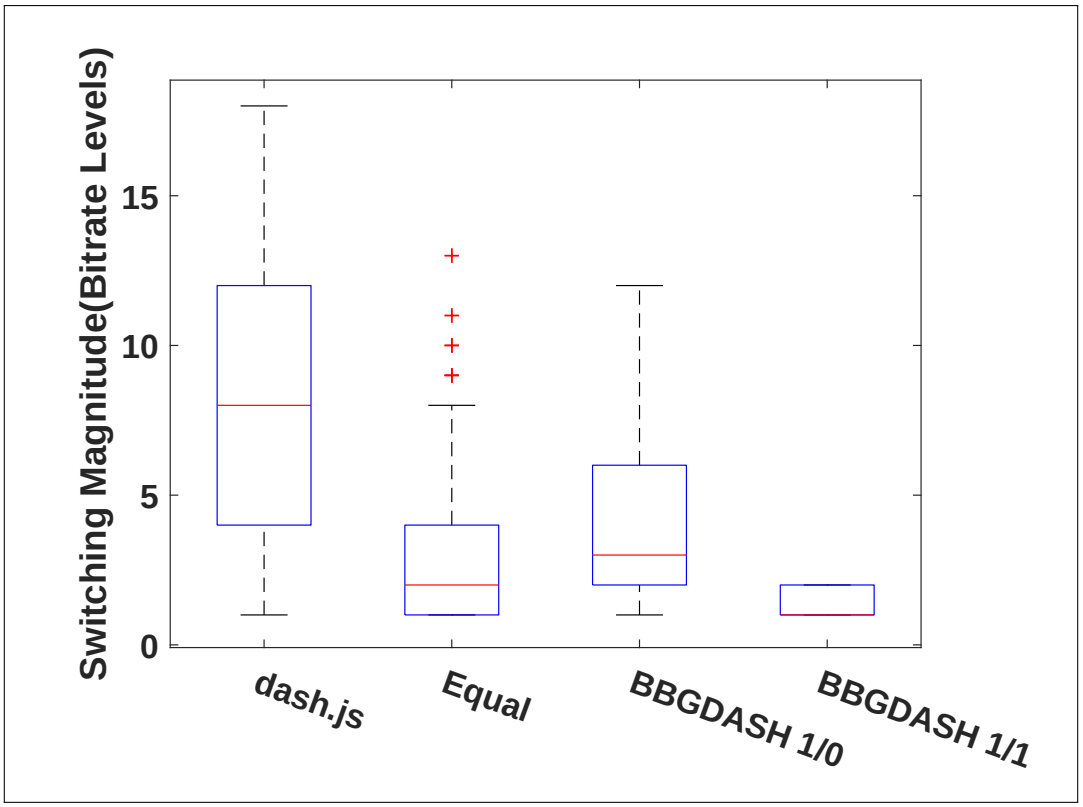


Figure 4.5
Amplitude of Bitrate Switching Events

utilisation with a high value of underutilisation index equal to ($M = 0.36$) for unassisted dash.js, ($M = 0.23$) for the upper-bound assisted dash.js, and ($M = 0.2$) for BBGDASH 1/0.

Normalised QoE and Fairness Level

The main objective of BBGDASH is to provide a scalable solution that fairly maximises the perceived QoE and efficiently utilises the available resources. Since the video contents are available in the CDN with varied maximum levels, equal QoE allocation is not always efficient. Accordingly, Figure4.7, and 4.8 compare the performance of the proposed approach against others in term of perceived Normalised-QoE and the distributed fairness among the different DASH players. As illustrated in Figure4.7, deploying BBGDASH 1/1 results in boosting the average perceived QoE to 0.72 from 0.34, 0.4, and 0.39 with dash.js, Equal, and BBGDASH 1/0 respectively. This value, however, represents the median QoE level among the different players requesting videos with disparate maximum bitrate levels, as depicted in section 4.4

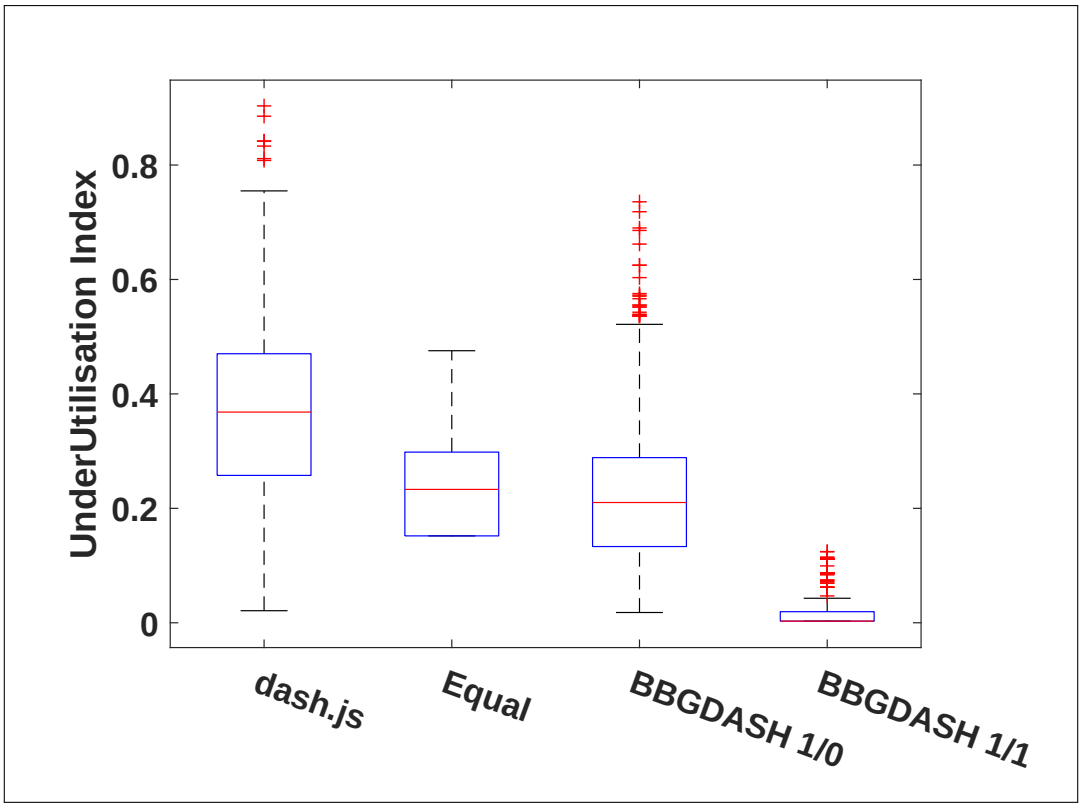


Figure 4.6
Underutilisation Index.

.

A similar improvement can also be observed when comparing the fairness achieved by the compared approaches, as shown in Figure4.8. The figure reveals that BBGDASH 1/1 increased the average value of the achieved fairness to 2.3, compared to 1.4, 1.7, and 1.8 for dash.js, Equal, and BBGDASH 1/0 respectively. The main reason behind this improvement is that BBGDASH 1/1 utilises the remaining bandwidth to boost the quality of the other players, which ends with maximising the allocation of QoE among players.

4.5.1 Summary

When using HTTP adaptive streaming, client-driven HAS applications locally optimise the adaptation decision without coordinating between each other, which may lead to an unfair QoE distribution among the existing clients. Solutions like QoE-centric approaches may harm the principle of HAS and generate scalability

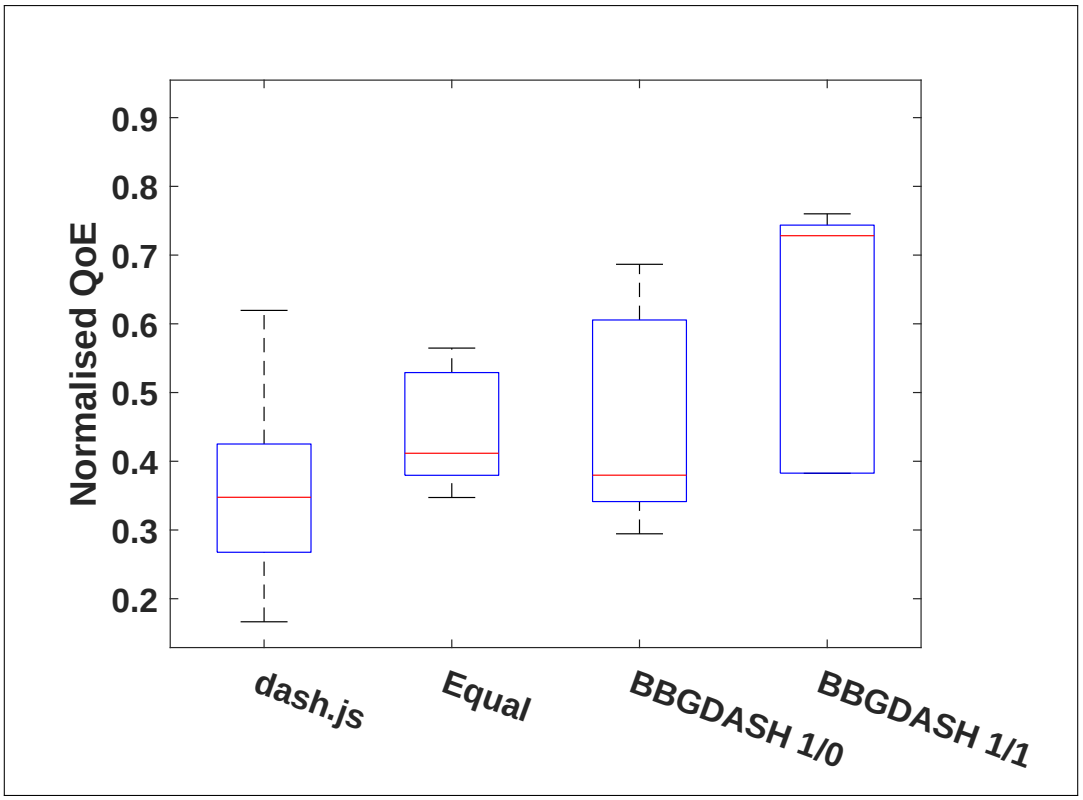


Figure 4.7

Normalised QoE

issues [37], as the adaptation decision is transferred to a specific element. To overcome the specific limitations of each approach, this chapter introduces a hybrid guidance mechanism that provides guidance to the client without moving the entire control logic to an additional entity or solely relying on the client-side decision. In order to provide a realistic evaluation of the proposed approaches and compare their performance with the other benchmarked approaches, an implementation of the SDN-based architecture is performed.

The experimental evaluation of a video streaming scenario with nine heterogeneous DASH players using a proof-of-concept implementation demonstrated the potential of the proposed approach compared with the other benchmarked solutions. In this specific scenario, BBGDASH outperforms the other solutions and increases network resource utilisation up to 36%, and QoE up to 38%, compared to the other approaches.

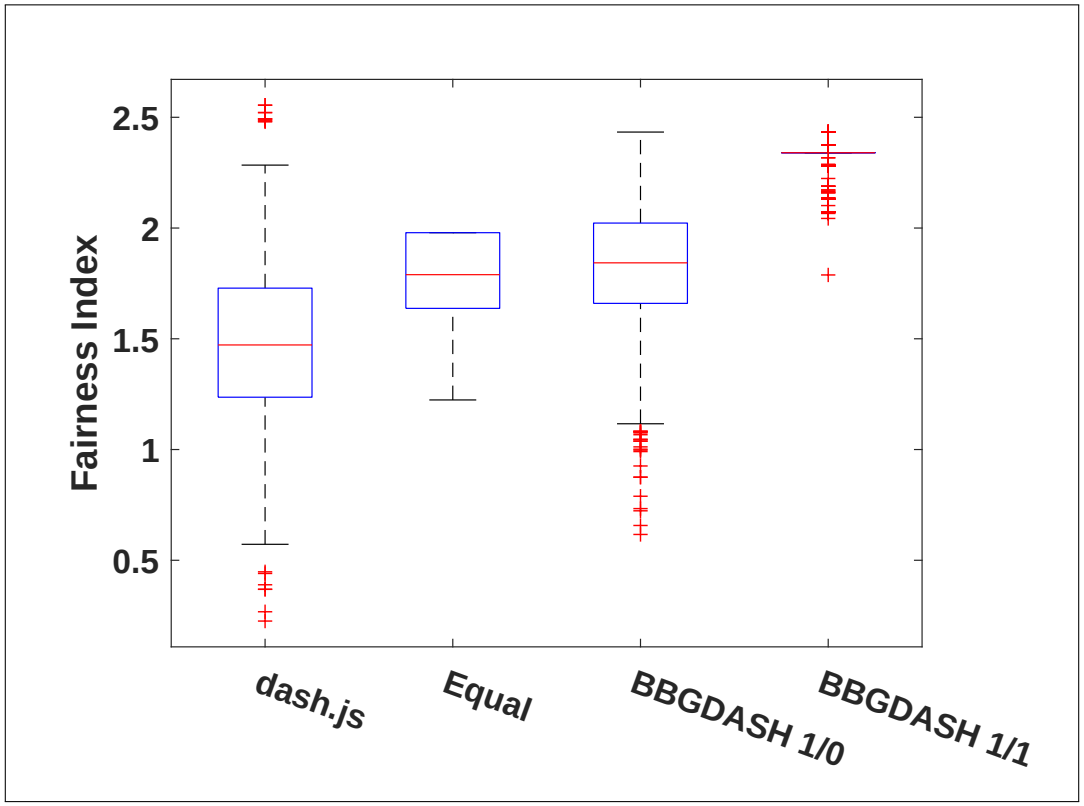


Figure 4.8

Fairness Level

Chapter 5

Bandwidth Prediction Schemes for Defining Bitrate Levels in SDN Enabled Adaptive Streaming

5.1 Introduction

Although BBGDASH is an efficient approach for video delivery within wired networks, deploying BBGDASH in a wireless network environment could result in sub-optimal decisions due to the high fluctuations in the wireless environment. Moreover, identifying the number of bitrate levels for each client in a wireless network condition is non-trivial and could result in a significant QoE degradation when the client is bounded with an incorrect set of bitrate levels. Nevertheless, because the network conditions are different, the bitrate guidance should be robust regardless of time, environment, and network operator, as well as the nature of the streamed content.

Many studies [150] [111] [92] demonstrate the impact of accurate bandwidth prediction on improving the performance of the adaptation algorithms. It has been shown that reliable bandwidth prediction can improve the adaptation algorithm performance , especially when combined with rate stabilisation functions at the client side.

To that effect, this chapter extends the work presented in Chapter 4, which demonstrated that the bounded bitrate guidance approach can appropriately allocate the available capacity based on the DASH clients' requirements. The network

component guides clients with a set of bitrate levels, enabling them to adapt the quality locally with the client-side adaptation algorithm. This chapter presents an intelligent streaming architecture (denoted BBGDASH$^+$) that adapts the guidance based on the network conditions. BBGDASH$^+$ [27] leverages the power of time series forecasting to facilitate accurate and scalable network-based guidance. Furthermore, Error-Based Boundary (EBB) and Confidence-Based Boundary (CBB) algorithms that exploit time series forecasting to identify the optimal boundaries of the requested bitrate are proposed. Extensive experiments are also conducted to evaluate the impact of the forecasting configuration parameters and network conditions on the guidance schemes.

The remainder of this chapter is organised as follows. Section 5.2 presents the architecture SDN-based intelligent streaming scheme (denoted BBGDASH$^+$), followed by the modeling of the proposed system along with boundary allocation schemes in Section 5.3. Section 5.4 details the characteristics of the different network conditions, after which the proposed approaches are evaluated within the testbed that's presented in Section 5.5. The experimental results of the different approaches are presented in Section 5.6, after which Section 5.7 summarises the findings of this chapter.

5.2 Proposed System Architecture for Bounded Bitrate Guidance for DASH

This section presents the proposed system architecture for the proposed bounded bitrate guidance. The proposed architecture (i.e. BBGDASH$^+$) extends BBGDASH [28]) by introducing the forecasting component, shown in the red colour, which deploys the two algorithms (EBB) and (CBB) in an attempt to identify the optimal boundaries of the requested bitrate levels. Nevertheless, BBGDASH$^+$ introduces a new communication channel, shown in the red colour, between the network component and the video server, which in turn allows the network component to send messages to the video server regarding the recommended bitrate levels for each DASH player. In line with the previous architecture, BBGDASH+ consists of three planes: a) the QoE management plane, b) the control plane and c) the data plane, as shown in

Figure 5.1. The description of each plane is provided below.

5.2.1 Data Plane

The data plane consists of a set of SDN-based forwarding devices, interconnected via wired/virtual connections. These devices, implemented in this work using Open vSwitch(OVS), are responsible for forwarding the network flows and allocating the available resources based on the rules applied by the control plane. The data and control planes communicate using the OpenFlow protocol [95] as the Southbound Interface (SBI) protocol.

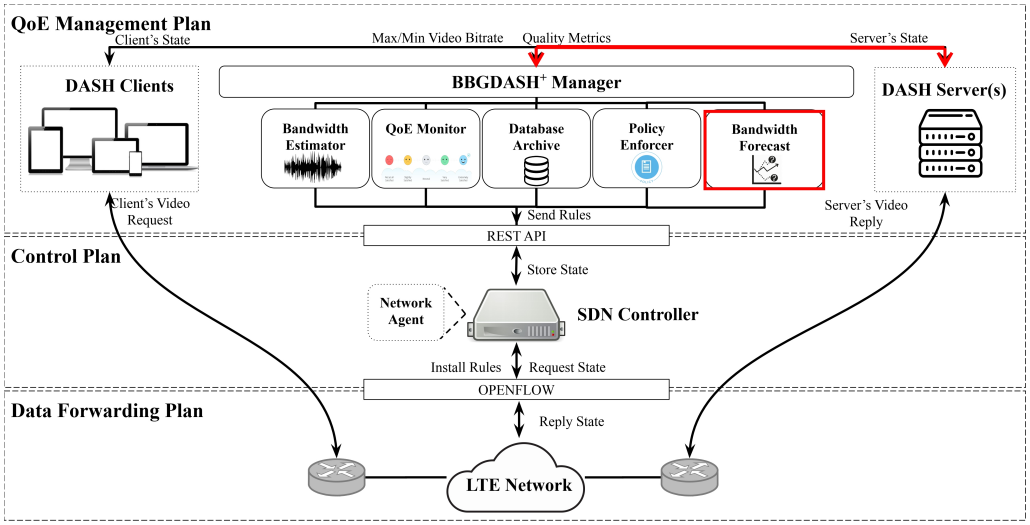


Figure 5.1

Proposed QoE-Driven Network-assisted Architecture for HTTP Adaptive Video Streaming

5.2.2 SDN Control Plane

The control plane provides a bridge between the data plane and the QoE management plane; this has been designed to support the delivery of video services and provide the QoE-based resource allocation per DASH client. The proposed architecture implements the Network Agent, which carries instructions or defined policies from the QoE management plane and translates these policies into a set of rules or actions on the data forwarding plane. Furthermore, Restful API [148] has been

employed to provide communication between the QoE management plane and the control plane, with OpenFlow acting as the southbound interface protocol.

5.2.3 QoE Management Plane

The QoE management plane consists of six modules: BBGDASH$^+$ manager, Bandwidth Estimator, QoE Monitor, Database Archive, Policy Enforcer, and Bandwidth Forecasting. The details of each component in the proposed architecture are presented below.

BBGDASH$^+$ Manager

The BBGDASH$^+$ manager is responsible for computing the optimal set of bitrate levels per DASH player. The manager communicates with the other components of the QoE management plane in order to obtain the required information (e.g., number of active DASH players, available resources, etc.) in order to allocate network resources and guide DASH players. It is aware of the incoming DASH flows based on client requests and guides the video players by sending the optimal boundary levels of the requested bitrates.

Bandwidth Estimator

The Bandwidth Estimator requires previous knowledge of the network path's link capacity to estimate the available bandwidth. In the proposed architecture, the control plane queries the forwarding devices in the data plane frequently to obtain the network statistics. These collected network statistics are then smoothed and sent to the Database Archive, where they are used as input for forecasting purposes.

QoE Monitor

The QoE monitor module performs real-time measurement for the perceived QoE. This module considers the devices, video, and other QoE related metrics – including average bitrate, bitrate switching amplitude, and stall duration – used to calculate the end-user QoE. Furthermore, this module is sufficiently flexible to allow the deployment of any other QoE metric-based model used to measure the perceived QoE for the end user.

Database Archive

The database module stores information related to the number of DASH clients, streaming states, configuration parameters, estimated bandwidth and the QoE metrics of video streaming sessions. It also provides the required information to the BBGDASH^{+} manager module and the Forecasting module, which is then used as inputs to compute the suitable set of bitrate levels. As shown in Figure 5.1, based on the feedback loop mechanism for configuration selection, the database is used to store network statistics and DASH flow rules that are currently active in the streaming sessions.

DASH server and client entities

The DASH server hosts DASH content, encoded with FFmpeg into multiple versions and segmented using GPAC MP4Box [9] into small chunks. Each video is accompanied by a manifest file (i.e. Media Presentation Description (MPD)) that describes its characteristics, such as video periods as adaptation sets. On the client side, a modified dash.js player embeds the proposed guidance schema. In addition, the MPEG Server and Network Assisted DASH (SAND) architecture [130] are utilised to enable the client to send the QoE-related metrics (i.e. content and device properties) and receive the optimal guidance to/from the proposed BBGDASH^{+} agent.

Policy Enforcer

This module applies the BBGDASH^{+} recommendation at the application and network levels. The application's level action is implemented by sending the bitrate level range for each DASH player, while the network level action is applied by dynamically allocating the network resources. This module also sends messages about each DASH client levels to the video server, which in turn prevents the greedy clients from requesting higher bitrate levels than those allocated by BBGDASH^{+} manager.

5.3 System Model and Problem Formulation

5.3.1 System Model

The system model is represented as an undirected graph $\mathcal{G} = (\mathcal{X}, \mathcal{Y})$, in which $\mathcal{X} = \{P, F, A, S\}$ represents the set of nodes and $\mathcal{Y}$ denotes the set of links between the nodes. The set of nodes in the proposed architecture includes a number of subsets that encompass a number of DASH players $p_i \in P$, one DASH server S, forwarding devices F, and the SDN controller A. Each player p_i has at least one link $y \in Y$ that connects it to the access node $f \in F$. It has been proposed that the only network bottleneck is located in the access network, with a total estimated capacity of BW_e, as well as an estimation for the cross traffic BW_c at time t. Consequently, the end-to-end path residual bandwidth for streaming HAS traffic in the same time slot will be BW_{HAS}, as shown in (5.1):

$$BW_{HAS} = BW_e - BW_c \tag{5.1}$$

Each player $p_i \in P$, $i = [1, \ldots, N]$ has a specific set of requirements for screen resolution r_{pi} and plan subscription ST_{p_i} (i.e. normal, bronze, silver, gold, platinum), and may also request a video $v_{p_i} \in V$ that has a distinct set of bitrate representations $L^{v_{pi}}$.

Given this context, enforcing the same bitrate allocation for all players may result in an unfair distribution and the inefficiency of both QoE and network resource allocation. In order to meet the end-user QoE requirements and achieve a high level of fairness among players requesting different videos, such that each video has a type ({animation, sport, news, movie, *etc.*}), set of resolutions ({360p, 480p, 720p, 1080p, *etc.*}), and various bitrate levels. The network-assisted approach was designed with reference to the concept of *max-min fairness* [81].

The main objective is to maximise the minimum quality across the p_i players. The objective function f is described as follows:

$$f = \begin{cases} \max \left(min\ l^j_{p_i} \in L^{v_{p_i}} \right) \\ s.t\ \sum_{i=1}^{N} BW_{p_i} \leq BW_{HAS}, \\ \quad BWp_i \geq 0, l^j_{p_i} \in L^{v_{p_i}} \\ \quad \forall p_i \in P, i = [1, \ldots, N], j = [1, \ldots, M_{p_i}], \end{cases} \tag{5.2}$$

where BW_{p_i} is the bandwidth allocated for p_i, $l^j_{p_i}$ is the bitrate selected by p_i, and M_{p_i} is the total number of bitrate levels for the video v_{p_i}.

5.3.2 ARIMA-based Bandwidth Forecasting

Future bandwidth prediction is a time series forecasting problem [49]. In this problem, for a given timeslot, the objective of the prediction model (μ) is to accurately find the predicted bandwidth when given a set of previous bandwidth measurement samples. Formally, given a set of n time series observations at elapsed time t, denoted as y_t, a future predicted value of time series for the next h steps (horizon), denoted $\hat{y}_{t+h}$, is defined as follows:

$$\hat{y}_{t+h} = \mu(y_t, y_{t-1}, y_{t-2}, \ldots, y_{t-n}), \tag{5.3}$$

where n is the number of observations.

In this context, the primary challenge is to choose the model that provides the highest forecasting accuracy. Several studies [92], [150] have acknowledged that statistical approaches tend to have higher forecasting accuracy than machine learning models [91]. Moreover, Continuous Learning (CL) or Lifelong Machine Learning (CML) [106] is an emerging approach that has recently attracted the attention of many researchers. This study integrated the autoregressive integrated moving average (ARIMA), a classical and universal statistical modelling tool, with CML-based strategy for predicting $\hat{y}_{t+h}$, given their abilities to handle the non-stationary time series data by applying the differencing method and continuously updating the model.

When attempting to model ARIMA processes, it has been considered that the variable y_t is a set of the network bandwidth measurement samples. Accordingly, the first part of the ARIMA model is an autoregressive (AR) process of order p for the number of time lags y_t, as follows:

$$\hat{y}_{t+h} = c + \phi_1 y_t + \phi_2 y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t, \tag{5.4}$$

Here, c is a constant, ϕ_j are the coefficients or parameters of the moving average part, and ε_t is the error term. Furthermore, the AR model can be rewritten as a

general expression by using the backshift notation:

$$(1 - \phi_1 B - \cdots - \phi_p B^p)\ \hat{y}_{t+h} = c + \varepsilon_t. \tag{5.5}$$

Another process involved in the ARIMA model is the moving average (MA) of order q, which is a linear combination of the current white noise term and the q of the most recent past white noise terms. It can be defined as follows:

$$\hat{y}_{t+h} = c + (1 - \theta_1 B - \cdots - \theta_q B^q)\ \varepsilon_t, \tag{5.6}$$

where c is a constant and ε_t are the error terms.

The AR and MA models can be integrated concurrently with differencing the time series to yield a wide variety of effects. This combination leads to an *ARIMA(p,d,q)* model, which can be written in backshift notation as:

$$\begin{aligned}(1 - \phi_1 B - \cdots - \phi_p B^P)\ (1 - B)^d\ \hat{y}_{t+h} = \\ c + (1 - \theta_1 B - \cdots - \theta_1 B^q)\ \varepsilon_t,\end{aligned} \tag{5.7}$$

where ϕ_i, θ_i are parameters to be determined.

The first 95% confidence interval for the predicted data can be given as follows:

$$\alpha = \hat{y}_{t+h} \pm 1.96\hat{\sigma}, \tag{5.8}$$

where $\hat{\sigma}$ is the standard deviation of the residual errors.

The root mean square error (RMSE) was used to evaluate the accuracy of the forecasting model and the goodness of its parameters, as it indicates the squared error between the predicted $\hat{y}_{t+h}$ and the observed value(s) for the same throughput levels. Moreover, the MAPE is used to assess the prediction for different throughput levels.

In this context, it has been considered that the width of the confidence bands and the width of the error bands, given the confidence bands, bound 95% of the true distribution of the data, while the error bands indicate the difference between the real throughput levels and the measured error itself. Therefore, decreasing the width of the bitrate bands may result in accurate guidance.

5.3.3 Throughput Prediction

The two key elements of the proposed forecasting methodology are as follows: to find the best fit of the ARIMA model that achieves the highest accuracy, and to find the optimal tuning for the prediction configurations that minimises the width of the confidence interval bands and leads to a better bitrate selection. The *auto.arima*() Python library function was used to determine the optimal online fit of the ARIMA model. The function is based on a variation of the Hyndman-Khandakar algorithm [66], which integrates unit root tests, maximum likelihood estimation, and cross-validation techniques. The algorithm begins by identifying the number of differences d required to generate stationary data. After differencing the data d times, the algorithm chooses values of p and q by minimising the Akaike Information Criterion (AIC), which can be written as follows:

$$\text{AIC} = -2\log(L) + 2(p + q + k + 1), \tag{5.9}$$

Here, L is defined as the likelihood of the data, while k is the number of estimated parameters in the model. To choose the p and q that minimise the AIC (5.9), the algorithm utilises the stepwise search to navigate the model space rather than considering every possible combination of p and q.

Accurate bandwidth prediction is essential to the provision of proper guidance for DASH players. The presented methodology starts with splitting the traces into two parts. The training part comprises 60% of the trace and is used to create the main prediction model; the other part is used for online prediction and validation. The prediction methodology retrains the prediction model with the new observed values every h seconds. Although the prediction horizon h and the number of predicted values can be used as input variables for the model, in this study, h has been fixed and remains proportional to the duration of the requested video chunk.

5.3.4 Video Bitrate Boundary Identification

The main objective of the proposed approach is to identify the optimum set of bitrate levels that should be used by DASH players to adopt the quality locally. To achieve this, two algorithms are introduced: namely, Confidence-Based Bounding (CBB) (Algorithm 2) and Error-Based Bounding (EBB) (Algorithm 3), both of

which exploit the power of time series forecasting to identify the optimal boundaries of the requested bitrate.

For each step i, CBB begins by forecasting the throughput $\hat{y}_{t+h}$ for the horizon h, $h > 0$, then computes the confidence interval value (α), which identifies the initial boundaries of the requested bitrate levels. In the next step, the algorithm maps the continuous values of the computed boundaries into discrete bitrate values based on Algorithm 4 and ends by sending the computed boundaries to each DASH player.

Algorithm 2 Confidence-based Bounding (CBB)

1: **for each** *step* i **do**

2: Forecast the throughput $\hat{y}_{t+h}$ using (5.7).

3: Find the initial boundaries using (5.8).

4: Quantise the initial boundaries to the discrete video bitrate boundaries: Call algorithm 4

In Algorithm 3, step 3 has been replaced; this step now defines the initial boundaries based on the prediction error rather than the confidence interval.

Algorithm 3 Error-based Bounding (EBB)

1: **for each** *step* i **do**

2: Forecast the throughput $\hat{y}_{t+h}$ using (5.7).

3: Find the initial boundaries as follows:

$$\alpha = \hat{y}_{t+h} \pm RMSE \tag{5.10}$$

4: Quantise the initial boundaries to the discrete video bitrate boundaries: Call algorithm 4

5.3.5 Perceptual Quality and Cluster Identification

Video perceptual quality measurement has a non-linear relationship with bitrate [121, 48, 36]. To embed this in the proposed approach, the bitrate-to-perceptual quality mapping function from [37] was adopted; this function takes three features of a video streaming session (device resolution, content type, and service plan type)

and then maps all of them into one common SSIMplus-based space (the Structural SIMilarity plus index) [113]. With this model, the existing BBGDASH$^+$ players can be clustered into five non-overlapping clusters, denoted as $Cl = \{Cl_1, \ldots, Cl_5\}$. Hence, clustering players into a set of clusters helps BBGDASH$^+$ to send per-cluster rather than per-client bitrate recommendations. This in turn enables the solution to reduce the complexity of the optimisation algorithm. It is notable that the clustering identification is enabled only if there are many BBGDASH$^+$ players in the network.

5.3.6 QoE-driven Quality Optimisation

To achieve a fair QoE allocation among DASH players and an efficient resource utilisation, a dynamic programming based algorithm is proposed that provides each player with the optimal bitrate guidance. The proposed algorithm 4 provides fair QoE distribution and efficient resource allocation among different players with different requirements that may request videos with heterogeneous representation levels.

The algorithm begins by initialising the input variables, namely N, $L^{v_{p_i}}$, BW, Cl, N_{cl_k}, $l_{cl_k}^{opt}$, and $\alpha k = [1, \ldots, Z]$, and $i = [1, \ldots, N]$, which are defined with other corresponding notations in Table 5.1. In line (2), the algorithm computes the scaling factor S_f for each cluster $cl_k \in CL$, since each cluster includes a set of players that have the same requirements. Based on S_f, the algorithm computes the initial optimal bitrate level $l_{cl_k}^{ini,p_i}$ for each player p_i in cluster cl_k (lines 3-4). Next, for each player p_i within cluster cl_k, the algorithm identifies $l_{p_i}^{\sigma}$ as the higher bitrate level less than or equal to $l_{cl_k}^{ini,p_i}$ to be stored in V_1 (lines (5-8). Lines (9-16) compute the required bandwidth for each player to stream with the next bitrate $l_{p_i}^{\sigma+1}$. The amount of bandwidth remaining after the initial allocation BW^{left} is calculated in line (17). Lines (18–24) fairly redistribute the remaining bandwidth among players in a way that ensures a fair QoE distribution and efficient resource utilisation. The algorithm ends with sending the minimum and maximum bitrate level for each player.

Table 5.1

Notation and Symbols Description.

Symbol	**Descriptions**
N	Total Number of DASH players
Cl	Set of clusters $cl_k, k = [1, \ldots, 5]$ of specific screen resolution r_{cl_k}
N_{cl_k}	Number of DASH players in cluster cl_k
$l_{cl_k}^{ini,p_i}$	Initial bitrate selection for player p_i in cluster cl_k
$l_{cl_k}^{opt}$	Optimal bitrate for cluster cl_k
S_f	Scaling factor
BW_{HAS}	Total available bandwidth for HAS players at time t
BW_{p_i}	Bandwidth consumed by player p_i
BW^{left}	Remaining bandwidth after the first allocation
$L^{v_{p_i}}$	The complete set of bitrate levels $[l_{p_i}^1 : l_{p_i}^{M_{pi}}]$ of video v_{p_i} requested by player p_i
$l_{p_i}^{diff}$	Difference between the current bitrate level and the next level of the video v_{p_i} requested by player p_i
h	Prediction horizon related to the duration of the requested video chunk
V_1	Maximum allowed bitrate set for the players
V_2	Minimum allowed bitrate set for the players
α	The set of the max. and min. bandwidth computed by the prediction alg.

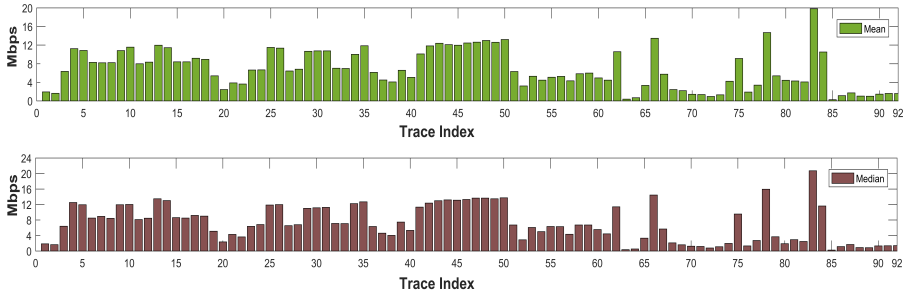
Algorithm 4 Bitrate Selection Algorithm

Input: $N, L^{v_{p_i}}, Cl, N_{cl_k}, l_{cl_k}^{opt}, \alpha$

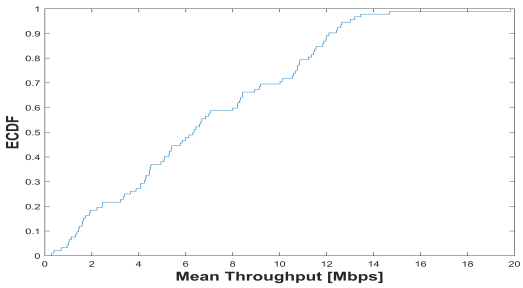
1: **for each** $j \leftarrow 1$ to 2 **do**
2: CALCULATE $S_f \leftarrow \frac{\alpha[j]}{\sum_{k=1}^{5} N_{cl_k} * l_{cl_k}^{opt}}$
3: **for each** *cluster* $cl_k \in Cl$ **do**
4: $l_{Cl_k}^{ini,p_i} \leftarrow l_{cl_k}^{opt} * S_f$
5: **for each** *cluster* $cl_k \in Cl$ **do**
6: **for each** *player* $p_i \in P$ **do**
7: FINDMAX$(l_{p_i}^{\sigma} \in L^{v_{p_i}}) \mid l_{p_i}^{\sigma} \leq l_{Cl_k}^{ini,p_i}$
8: INSERT$(l_{p_i}^{\sigma})$ *into* $V_1[]$
9: **for each** *player* $p_i \in P$ **do**
10: **if** $l_{p_i}^{\sigma} \neq l_{p_i}^{M_{p_i}}$ **then**
11: FIND$(l_{p_i}^{\sigma+1})$
12: $l_{p_i}^{diff} = (l_{p_i}^{\sigma+1}) - (l_{p_i}^{\sigma})$
13: **else**
14: $l_{p_i}^{\sigma+1} = \infty$
15: INSERT$(l_{p_i}^{\sigma+1})$ *into* $V_2[]$
16: INSERT$(l_{p_i}^{diff})$ *into* $V_3[]$
17: CALCULATE$(BW^{left}) = \sum_{i=1}^{N} l_{Cl_k}^{ini,p_i} - l_{p_i}^{\sigma}$
18: SORT$(V1)$
19: **while** $BW^{left} \geq$ MIN(V_3) **do**
20: **for each** $l \in V_1$ **do**
21: **if** $V_3[l] \leq BW^{left}$ **and** $V_2[l] \neq \infty$ **then**
22: $V_1[l] = V_2[l]$
23: $BW^{left} = BW^{left} - V_3[l]$
24: **else** CONTINUE()
25: RETURN(V_j)
=0

5.4 Network Traffic Characteristics

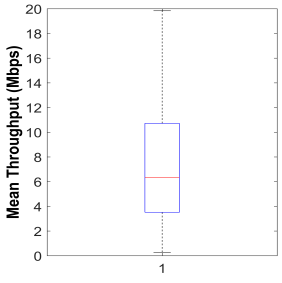
In order to evaluate the feasibility of deploying the bounding bitrate guidance under different network conditions, an appropriate dataset with varying network conditions is required to provide generic evaluations. With this in mind, the dataset of [96] has been used to conduct our evaluation. According to [96], the dataset used was generated from TCP throughput measurements in IEEE 802.11 wireless local area networks (WLANs) in different locations such as Berlin, Germany and Irbid, Jordan. The selected locations vary from very busy public hotspots like airports to less busy hotspots such as residential hotspots. The traces were collected on laptops running Ubuntu with default congestion control (i.e. TCP CUBIC). Moreover, the sender side is implemented via a set of servers located at the TU Berlin campus and in the Amazon cloud. The dataset consists of 92 traces of continuous TCP throughput measurements; each trace captures between 600 and 3600 measurements recorded with a resolution (i.e. measurement interval) of 1 second.



(a) Per-Trace Mean Throughput Measurement



(b) Empirical CDF vs Throughput of the Entire Dataset.



(c) Throughput Distribution of the Entire Dataset

Figure 5.2

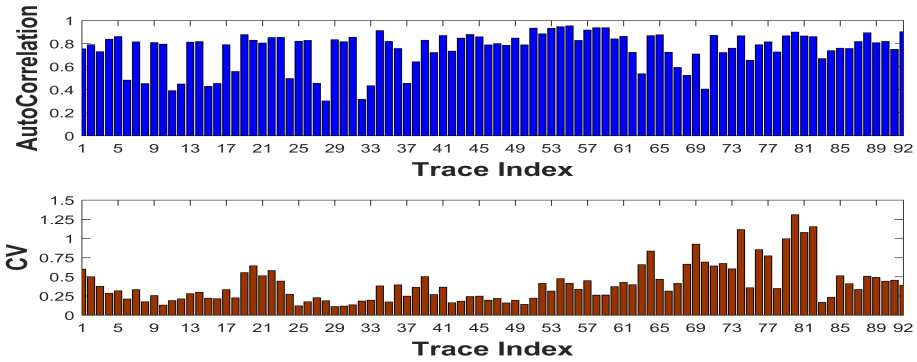
Network Throughput Measurements of The Entire Dataset

Figure 5.2a illustrates the mean and median throughput levels of each trace. It can be clearly seen from Figure 5.2b that the dataset covers a wide range of different network conditions, in which 50% of the traces lie between 6 and 8Mbps, while 90% of the traces cover the range from 1 to 1 Mbps to 3 Mbps. Furthermore, as Figure 5.2c indicates, the median level of all traces is equal to 6.3 Mbps. Figure 5.3a depicts the per-trace measurement for the autocorrelation and Coefficient of Variation (CV). The autocorrelation is a statistical representation of the similarity between a given time series and its lagged version over consecutive time intervals. Computing the autocorrelation operates in the same way as computing the correlation between two time series. However, autocorrelation uses the same time series twice (i.e. the other series is lagged for one or more time periods), computing the autocorrelation results with an output level between (+1) to (-1). An autocorrelation with a value of (+1) indicates that the series has a perfect positive correlation, which means that the increase of the time series will be followed by an increase in the next time period. On the other hand, an autocorrelation of (-1) denotes a perfect negative correlation (i.e. any time series increase will be followed by a proportionate decrease in the next time series). Accordingly, simple forecasting models, such as moving average or autoregression, are expected to improve in their achievements when the network traces have correlation levels close to 1. Figure 5.3b presents the distribution values of the autocorrelation at lag 1 among all traces. It would appear that 50% of the traces have an autocorrelation level ranging from 0.3 to 0.8, which provides an intuition about the various levels of the forecasting accuracy of the different traces.

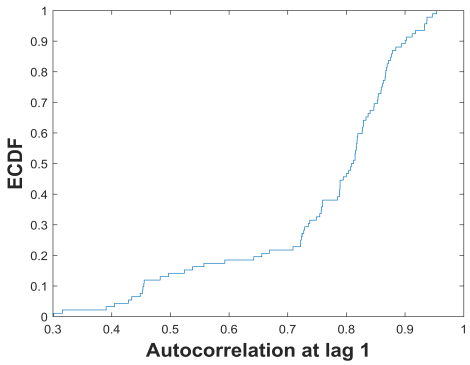
Coefficient of Variation (CV) is another statistical metric that indicates the variations of the different time series points from the mean, and is computed by dividing the standard deviation by the mean. The CV is an unbiased estimator of the degree of variation between different datasets, regardless of the mean value of each dataset. A low value of CV depicts that the time series is predictable, while a high value of CV results with lower forecasting accuracy.

Figure 5.3a indicates the CV level of each trace. It shows that trace 81 has the highest variation among the traces with a value of 1.3. By contrast, trace 29 has the lowest level (i.e. 0.11) of throughput variations among the different locations. Figure 5.3c also shows the distribution values of CV among all traces; it is evident

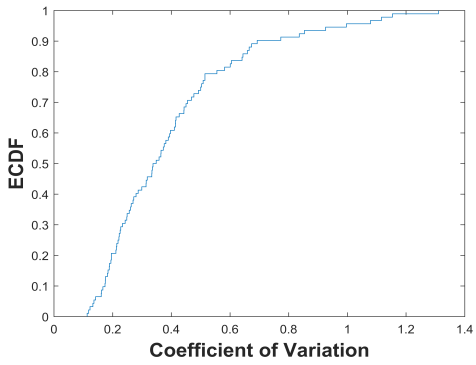
from the figure that 80% of the traces have a CV value less than 0.5.



(a) Per-trace Coefficient Variation and Autocorrelation At Lag 1



(b) ECDF vs the Autocorrelation at Lag 1of the Entire Dataset.



(c) ECDF vs Coefficient Variation of the Entire Dataset.

Figure 5.3

Coefficient Variation and Autocorrelation At Lag 1 of the Entire Dataset

Skewness and kurtosis are additional statistical metrics that explain the symmetrical curve and the peakedness/flatness of the normal distribution. Skewness measures the lack of symmetry of the given distribution; here, the data distribution can be negatively skewed, positively skewed, or symmetric. If the distribution has a positive skewness (i.e. a skewness level more than 0.5), then the data distribution will have a mean lower than the median with a flatter tail on the right side. On the other hand, negative skewness (i.e. less than -0.5) results in a mean level higher than the median, along with a long tail on the left side. This metric can also be a good indicator for tuning the adaptation algorithms to be more conservative or aggressive.

Similar to skewness, kurtosis is another statistical metric that describes the tails of the distributions and a measure of the outliers present in the distribution. This

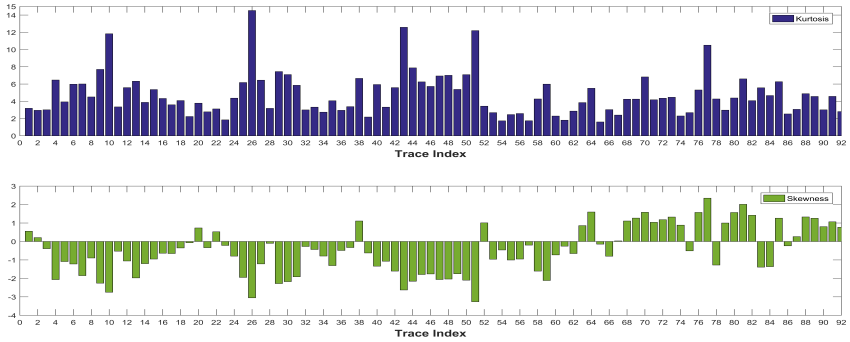
metric can also provide an intuition about the impact of the forecasting horizon on the RMSE accuracy. A high kurtosis value (i.e. above 3) means that the distribution has many outliers; moreover, low kurtosis (i.e. under 3) means that the distribution has a lack of outliers. Both skewness and kurtosis have significant impacts on the forecasting schemes, as they focus on extreme data rather than the average.

Figure 5.4a plots the per-trace level of skewness and kurtosis. It is evident from this graph that the collected traces are associated with different levels of symmetry. Furthermore, Figure 5.4b indicates that 50% of the dataset are negatively skewed, 30% are positively skewed, and 20% are symmetric. Accordingly, the aggressiveness of the adaptive video algorithm needs to be tuned based on the skewness level of the network conditions. In more detail, the ABR algorithm should be made more aggressive if the network experiences positive skewness and more conservative if the network skewness is negative. On the other hand, Figure 5.4c shows that about 70% of all traces have outlier measurements, which lead to a kurtosis level that exceeds 3. Therefore, long-term forecasting horizons should be applicable only for 30% of the dataset.

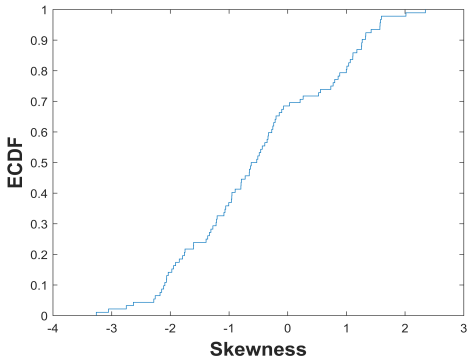
Resource outage is another feature of the wireless network conditions that occur when the network loses its resources. This can occur as a result of one of the following issues: power outages, operational/configurational error, hardware/software failure, or environmental issues. Outages are a focus of this analysis because they have a significant impact on video streaming performance. Figure 5.5a presents the per-trace outage distribution and duration. It can be clearly seen from the figure that some traces experience the outage phenomena. In more detail, about 35% of all traces experience an outage, with a distribution between 2–20, as depicted in Figure 5.5b. On the other hand, Figure 5.5c reveals that 70% of outage events have a mean duration over less than five seconds, while the maximum counted duration for an event is 32 seconds.

5.5 Experimental Setup

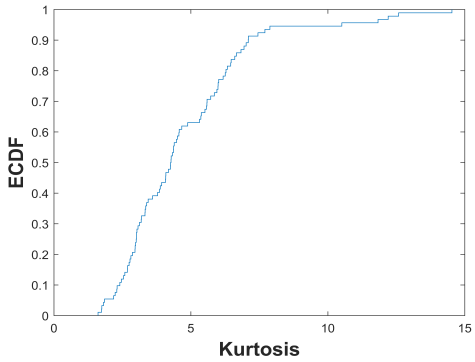
The architecture proposed in section 4.2 has been implemented in a testbed environment in order to investigate the performance of the proposed approaches. This



(a) Per-trace Kurtosis and Skewness Measurement



(b) Distribution of Skewness Measurement of the Entire Dataset



(c) Distribution of Kurtosis Measurement of the Entire Dataset

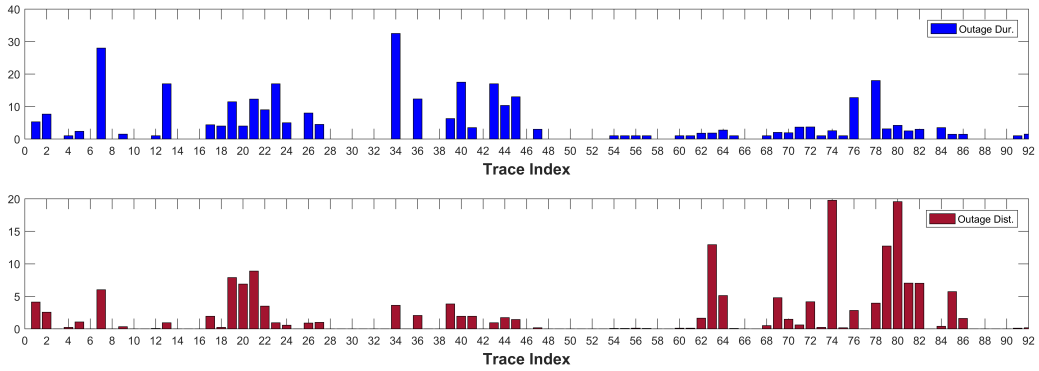
Figure 5.4
Kurtosis and Skewness Measurements of The Entire Dataset

section presents the testbed setup and explains the experimental methodology.

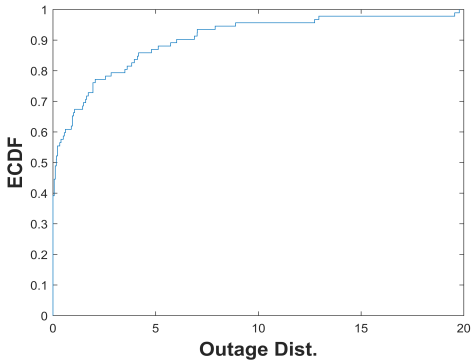
5.5.1 Evaluation Testbed

This set of experiments were conducted to assess the proposed SDN-based solution, which was implemented on three virtual machines (VMs) running Linux (Ubuntu 16.04 LTS) as shown in Figure 5.6. The testbed is divided into three planes: the data plane, control plane, and application plane. The data plane is implemented on Mininet[14] V2.3 network emulator and consists of two OpenFlow switches and a set of DASH players. However, an evaluation of the fairness of the proposed approach is outside of the scope of this paper.

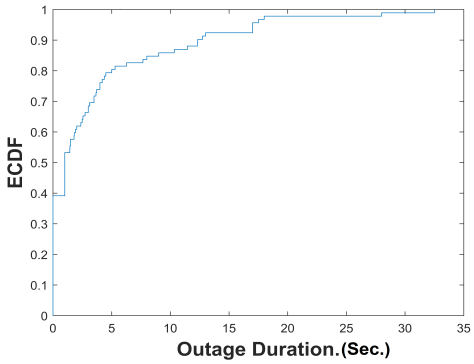
The control plane is implemented using the Ryu SDN controller [17], which is a Python-based SDN controller. Ryu also employs the OpenFlow v1.3 protocol as



(a) Per-trace Throughput Outage Measurement



(b) Distribution of Throughput Outage Measurement



(c) Distribution of Throughput Outage Duration

Figure 5.5
Throughput Outage Distribution of the Entire Dataset

the southbound interface to apply the required QoS configurations and periodically pull network statistics. The RESTful API has also been used as the northbound interface to provide communication between the control and management planes. In order to provide video access for the DASH clients and host the video content, an Apache server was attached to the Mininet network; specifically, the Big Buck Bunny video, which is 600 seconds long and encoded using FFmpeg at four different resolutions (360p, 480p, 720p, and 1080p) and a set of 20 bitrate levels using two passes. Each video is then segmented into sets of four-second chunks using GPAC MP4Box [10]; this duration was selected to make it proportional to the forecasting horizon.

At the client side, the dash.js player [7] was deployed, which is a JavaScript-based DASH player. Each DASH client was able to send the QoE-related metrics and receive optimal guidance to/from the proposed `BBGDASH`$^{+}$ agent (see Figure 5.1)

using a websocket channel between the players and the BBGDASH$^+$ agent.

WebSocket is selected over traditional methods (e.g., periodic polling and flash plug-in) to improve performance, as well as to significantly decrease network load and latency for real-time two way data communications. For archiving purposes, a MySQL database was deployed in the management plane for archiving the QoE and the network-related parameters.

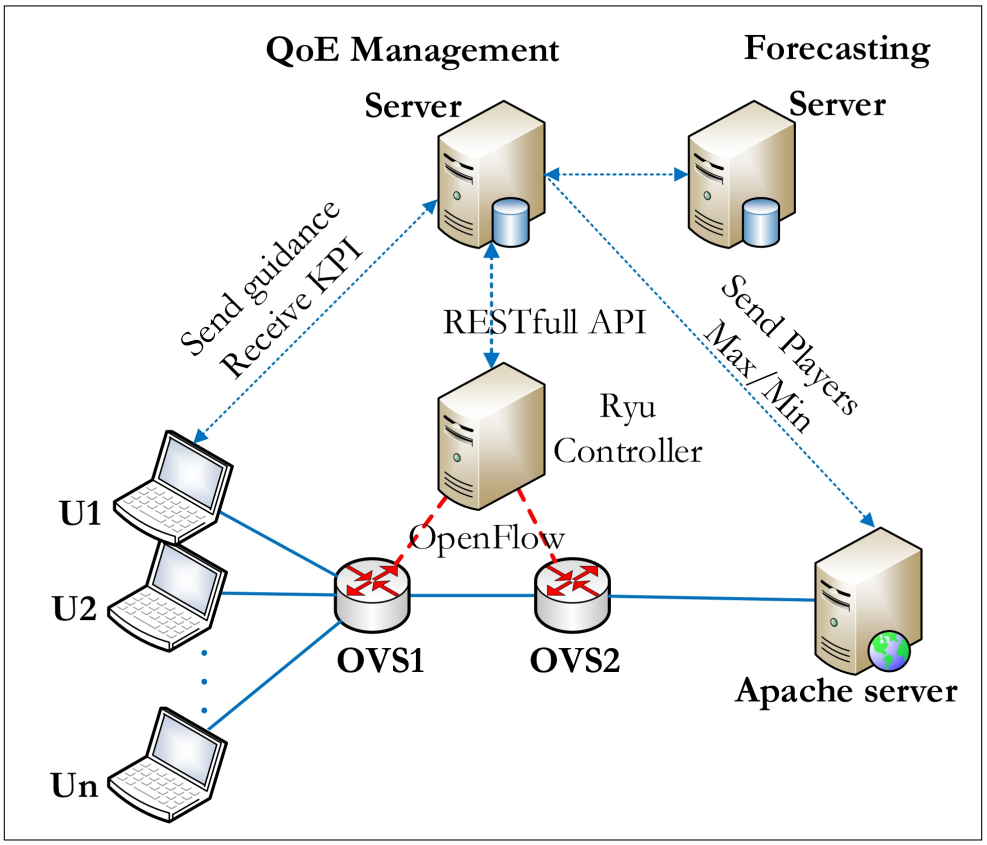


Figure 5.6

Experimental Testbed

5.5.2 Experiment Design

The implemented testbed was used to investigate the performance of BBGDASH$^+$, along with other benchmarked algorithms, under a wireless access network where the throughput is subject to considerable fluctuations. In order to replicate these network conditions, the link between OVS1-OVS2 was shaped based on real traces (as described in section 5.4).

***First Experiment*:** The first experiment aimed to investigate the impact of the

forecasting horizon, along with network features (i.e. section 5.4), on the accuracy of the prediction algorithm and how their proposed schemes (i.e. EBB and CBB) respond to the variations in network conditions and configurational parameters. To find the optimal configurations, the prediction algorithm was investigated under four different horizons (i.e. 2, 4, 8, and 12 seconds). Furthermore, each forecasting horizon was examined under different 83 network traces, which cover most of the network conditions.

***Second Experiment*:** The goal of the second experiment was to evaluate the stability of the received video and the efficiency of utilising the available resources when the DASH player streams video under wireless access network conditions without application or network assistance. In this experiment, the DASH player relied only on the client ABR algorithm (i.e. throughput-based) to adapt the quality of the received video to the estimated network conditions.

***Third Experiment*:** In the third experiment, the first network-based bitrate guidance approach (i.e. CBB) was investigated under a different set of configurational parameters (i.e. horizon) and variant network conditions to determine the efficiency of the proposed approach in providing efficient and stable video delivery. The potential of CBB was evaluated under three different horizons (i.e. 4, 8, and 12 sec.). Furthermore, for each of the examined horizons h, the CBB guidance approach was evaluated under 83 network traces.

***Fourth Experiment*:** Another network-assistance bitrate guidance approach (i.e. EBB) was evaluated in the fourth experiment to facilitate comparison with the other approaches. To ensure a fair comparison, the same configurational parameters and the network conditions of the third experiment were deployed in this experiment.

In order to evaluate the performance of the benchmarked approaches, a set of metrics have been adopted. The accuracy of the forecasting model and the goodness of the model parameters were measured using RMSE. Another metric is the band's width, which is identified as the difference between the upper and the lower band of the proposed approaches. For example, if $V_1 = 3$ Mbps and $V_2 = 2$ Mbps, then the band's width = 1 Mbps. Additionally, a set of metrics [121] designed for measuring the stability of the received video have been adopted, along with the perceived

QoE. In order to investigate the stability of the received video, the average bitrate, switching number and amplitude of the bitrate switching [121] were also used as performance metrics. The perceived QoE was calculated using the model presented in [145], which defines the QoE of video segments 1 through K using a weighted sum of the four QoE metrics (i.e. average bitrate, bitrate switching, video stalling, initial delay), which can be computed as follows:

$$QoE_1^K = \sum_{k=1}^{K} q(R_k) - \alpha \sum_{k=1}^{K} |q(R_{k+1}) - q(R_k)| - \omega \sum_{k=1}^{K} (\frac{d_k(R_k)}{C_k} - B_k) - \beta T_s \quad (5.11)$$

Furthermore, to obtain a standard measurement for QoE, the results of the QoE model could be normalised by dividing the computed QoE over the optimal QoE (QoE_{OPT}) value that can be perceived under the best conditions. In more detail, the optimal QoE (QoE_{OPT}) will be received when the client streams with the maximum available bitrate level and without any stalling or switching events.

$$nQoE = \frac{QoE}{QoE_{OPT}} \quad (5.12)$$

5.6 Experimental Results

This section investigates the feasibility of the proposed solutions by comparing them with other benchmarked solutions under different network conditions and configuration setups. The experiments we conduct are grouped into the following main categories: throughput prediction, client-based video streaming, CBB-based video streaming, and EBB-based video streaming (as presented in section 5.5.2).

5.6.1 Prediction Accuracy

Figure 5.7 presents the per-trace relative error for each of the investigated horizons. The relative-error percentage metric has been considered as a comparison metric to evaluate the accuracy of the forecasting algorithm under different network conditions. It can be observed from Figure 5.7 that the performance of the forecasting algorithm differs among the different network traces, as each trace is characterised by a unique set of network features (as discussed in section 5.5). For example, traces

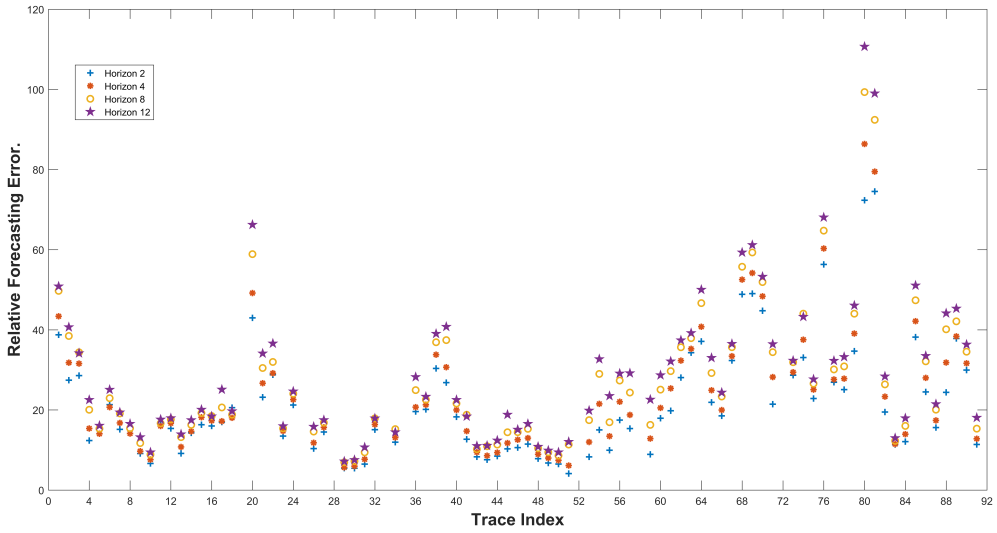


Figure 5.7

Per-trace Forecasting Accuracy

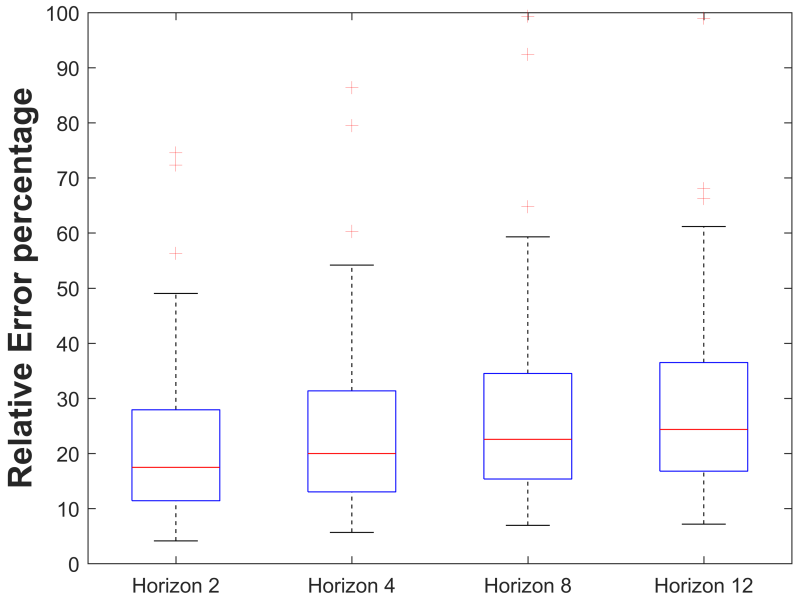


Figure 5.8

Forecasting Accuracy Distribution

29, 30, and 50 achieve a lower relative forecasting error within the investigated horizon (i.e. 2, 4, 8, and 12) with values of 4.1%, 5.4%, and 5.47% under ARIMA/2. On the other hand, traces 76, 80, and 81 count the highest forecasting errors, with values of 56%, 72%, and 74% under ARIMA/2. Figure 5.8 plots the distribution of forecasting accuracy for each of the investigated horizons; it can be seen from

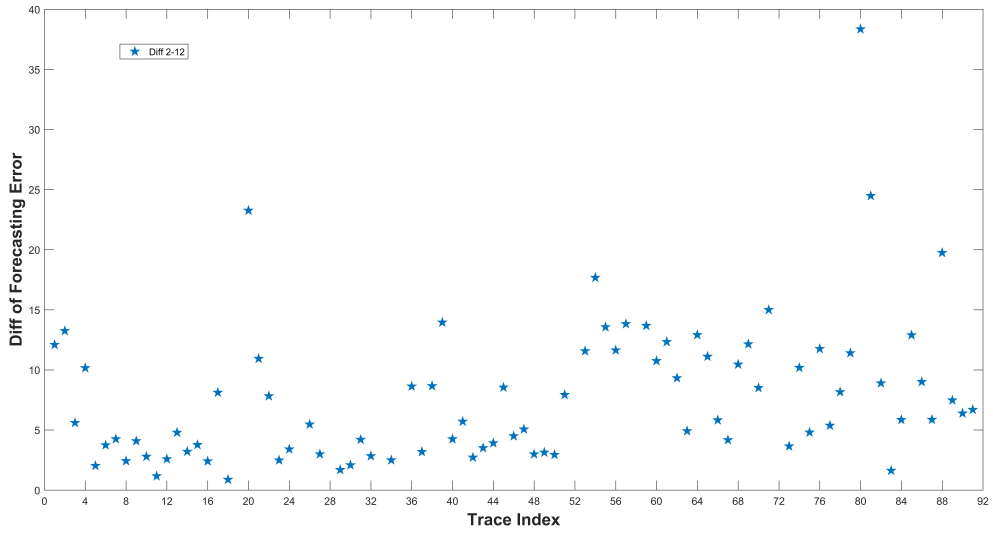


Figure 5.9

Per-trace Prediction Accuracy Differences Between Lower and Higher Horizon

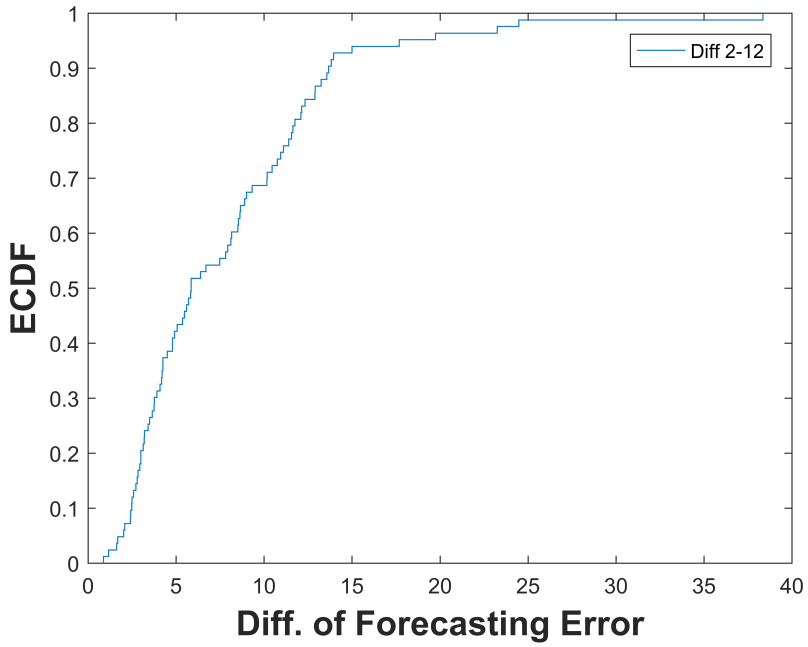


Figure 5.10

The Distribution of Prediction Accuracy Differences

this figure that the forecasting accuracy is proportional to the forecasting horizon. However, the impact of the horizon also differs based on network conditions. Figure 5.9 indicates the relative forecasting error difference between the lowest and highest horizons (i.e. 2 and 12). It shows that traces 18, 11, and 83 have the lowest difference between the forecasting error of horizons 2 and 12. Moreover, traces 20, 81,

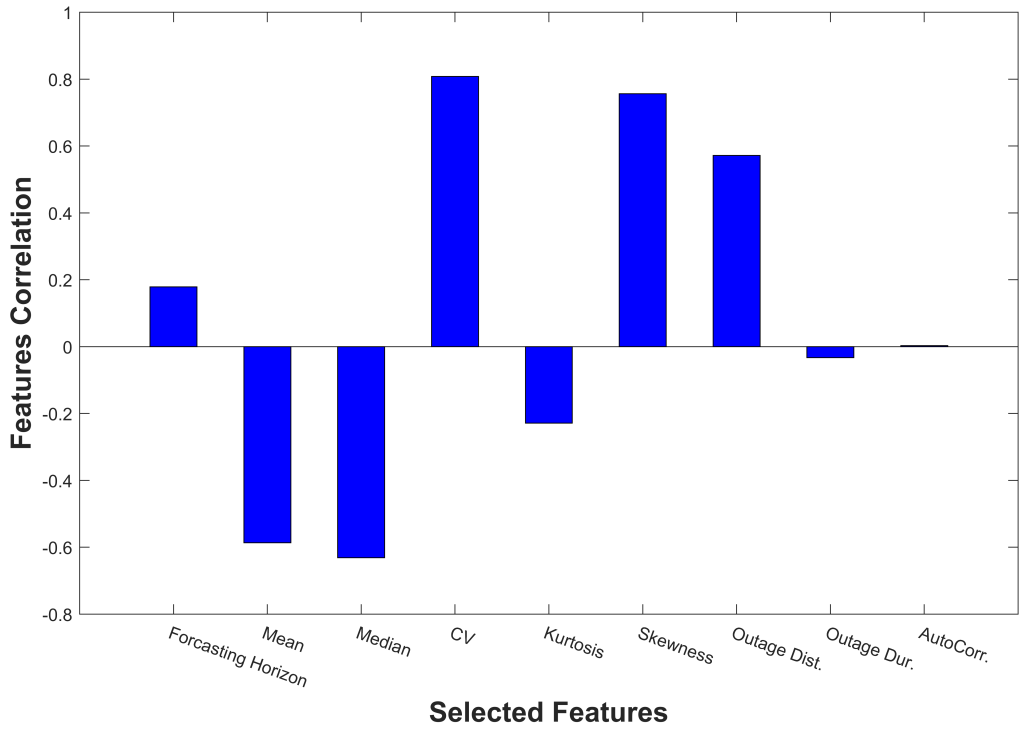


Figure 5.11

The Correlation Between Network Features and Forecasting Error

and 80 record the highest relative forecasting error differences between the lowest and highest horizons.

To understand the behaviour of the forecasting algorithm among the different network conditions, Figure 5.11 presents the correlation of the different features with the forecasting error. It can be seen that some network features (i.e. horizon, CV, skewness, outage distribution, and autocorrelation value at lag 1) are positively correlated with the forecasting error, while the other investigated network features (i.e. mean, median, kurtosis, and outage duration mean) are negatively correlated with the forecasting error.

CV and skewness are the network features most positively correlated with the forecasting error, with values of 0.8 and 0.75 respectively. By contrast, mean and median are the features most negatively correlated with the forecasting error. In order to explain the performance of the forecasting algorithm on the mentioned network traces, a closed observation is required to explore the per-trace features. It is necessary to examine the traces that achieve the minimum and the maximum forecasting error to assess their network features. For example, trace 18, which

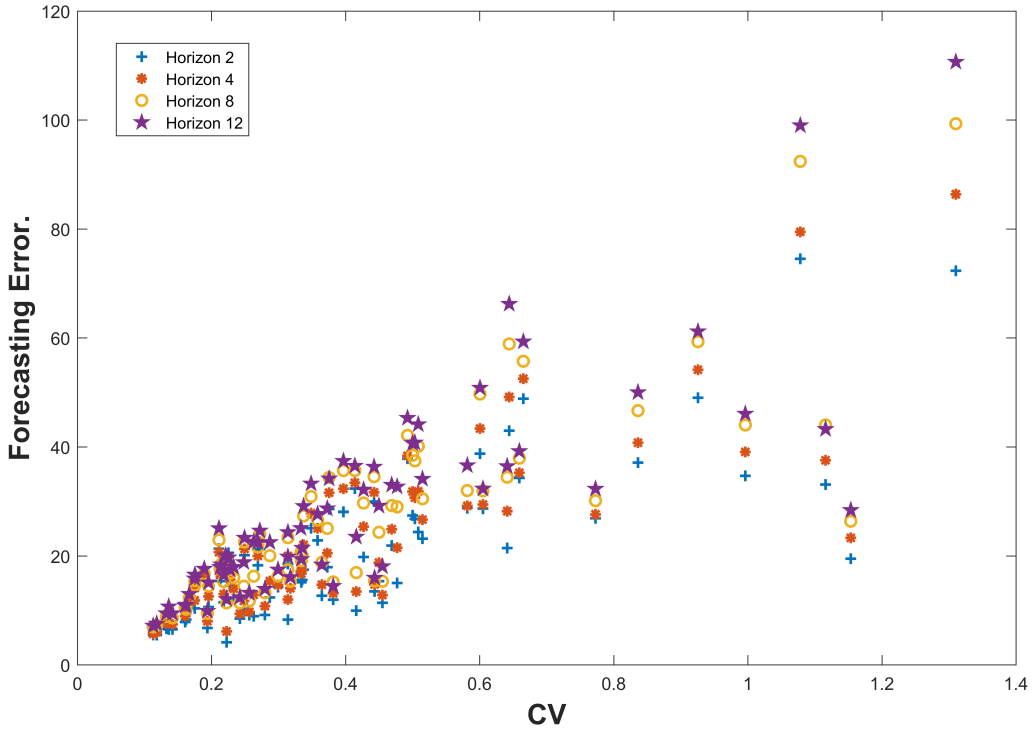


Figure 5.12

Forecasting Error Versus Throughput Coefficient of Variation (CV)

achieves the lowest forecasting error, has CV and skewness values of 0.11 and -2.2 respectively, while trace 76 (with the highest forecasting error) has the values of 0.85 and 1.5 for the same features.

Further analysis of the impact of the throughput coefficient of variation (CV) on the accuracy of the forecasting algorithm is presented in Figure 5.12. It can be observed from Figure 5.12 that a clear relation between throughput CV and forecasting accuracy exists; this is consistent with intuition, as the higher the throughput variation, the more challenging it is to provide high forecasting accuracy. A closer look shows that the forecasting algorithm achieves a relative forecasting error of value between 5-20% when the throughput CV is less than 0.2, while a higher throughput CV value results in a higher forecasting error value. The correlation between the network throughput variation and the forecasting error provides an interesting conclusion for considering the network throughput variation as a network differentiator. Therefore, the BBGDASH^{+} needs to be tuned conservatively (i.e. run under a lower forecasting horizon) when the network throughput CV has a value higher than 0.2.

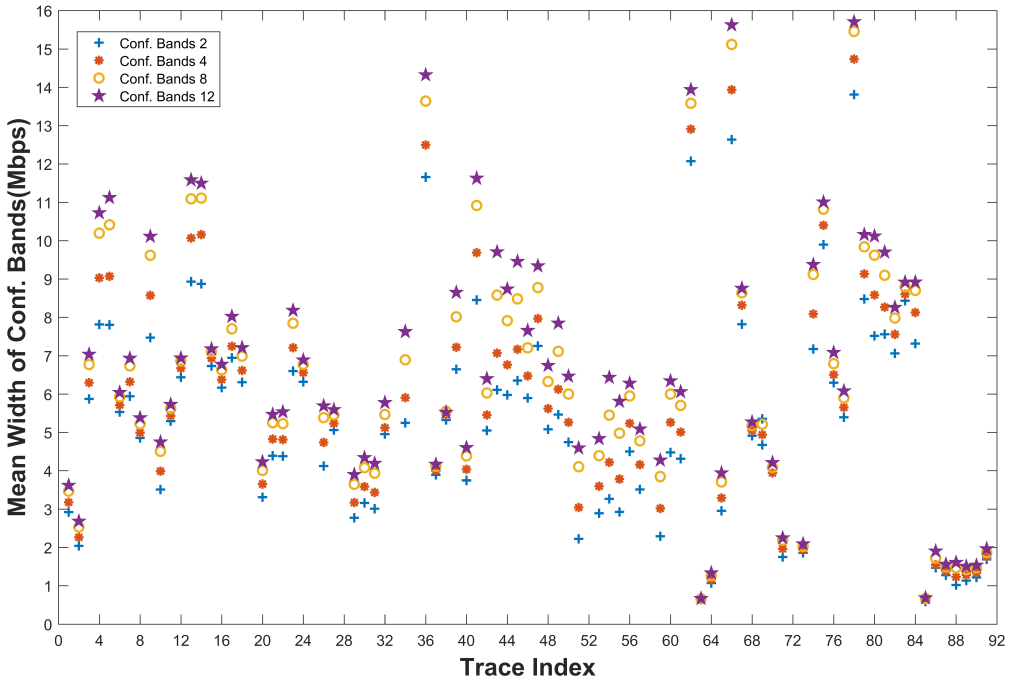


Figure 5.13

Per-trace Mean Width of Confidence Bands

5.6.2 Mean Width of Confidence Bands

Figure 5.13 shows the mean width of the confidence interval bands of the different horizons against the trace number. The width of the confidence bands represents the difference between the upper and lower confidence interval bands, as computed in algorithm 2. It can be seen from Figure 5.13 that the bands' width of the investigated horizon/traces is related to both the network conditions and the conducted forecasting horizon.

Figure 5.14 further explores the impact of each of the network features on the width of the confidence bands by plotting the degree of correlation between each network feature and the bands' width of the confidence interval. It can be observed that the confidence interval is more sensitive to the network features than the forecasting error. Furthermore, mean, median, and kurtosis are negatively correlated with confidence width. On the other hand, other investigated features (i.e. forecasting horizon, CV, skewness, and outage dist.) are positively correlated with the width of the confidence bands. Nevertheless, different features have different levels of correlation with regards to the width of the confidence intervals. CV and Skew-

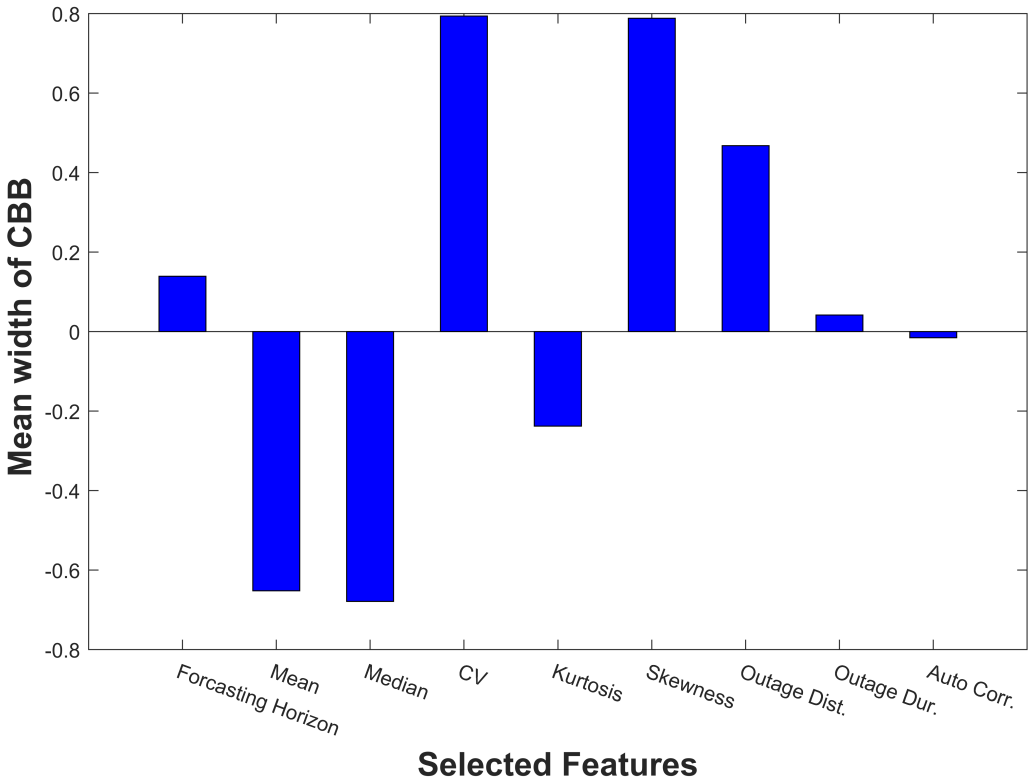


Figure 5.14

Correlation Between Network Features and Confidence Interval

ness are the features most positively correlated with the confidence interval width, with the correlation value for these features being 0.79 and 0.78 respectively. Moreover, the mean and median have the highest negative correlation among the other features. The impact of the forecasting horizon is lower than the impact of other network features, and it has different effects on the different network traces.

Figure 5.15 provides an initial intuition about the impact of confidence width on the perceived QoE by showing the number of filtered bitrate levels offered for each DASH player based on the CBB Scheme. A lower number of bitrate levels leads to more accurate bitrate guidance at the client side. However, bounding the player with the wrong number of bitrate levels may result in a negative impact on the perceived QoE. Therefore, the bounding scheme should provide a reasonable number of bitrate levels by taking the nature of the network conditions and video encoding level into consideration. As outlined in section 5.5.4, algorithm 2 (Confidence-Based-Bounding (CBB)) is used to identify the number of bitrate levels. Applying that algorithm for the different network conditions yields the results depicted in Figure

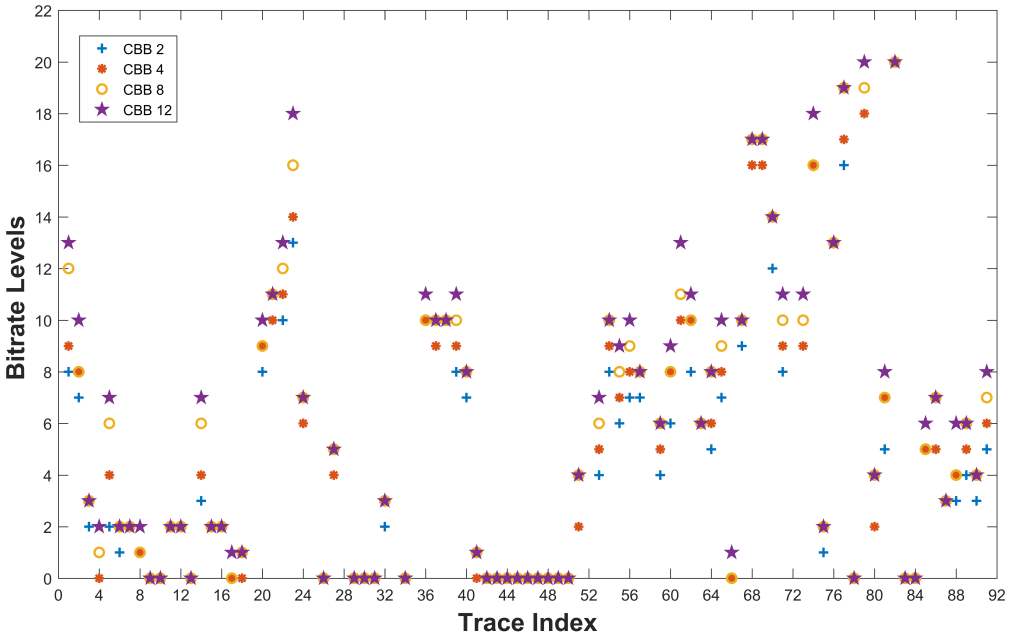


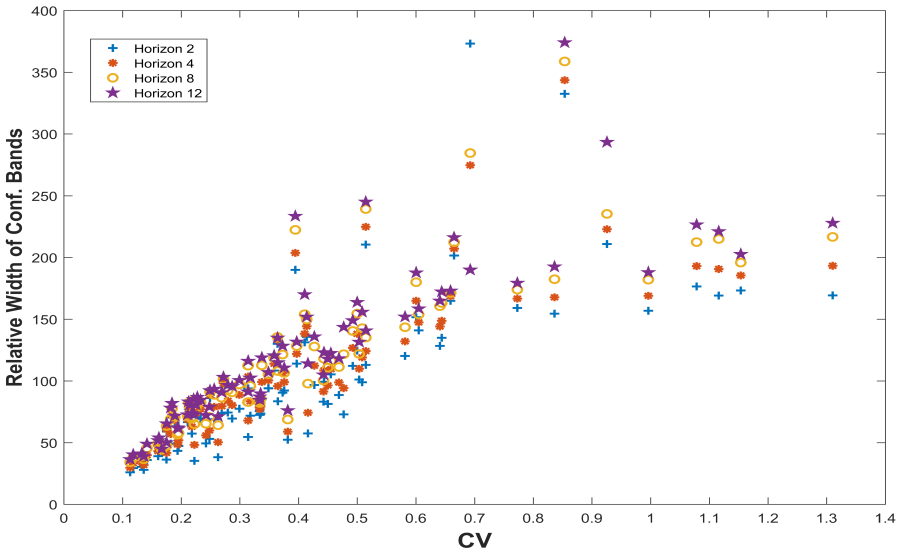
Figure 5.15

Per-trace Number of Bitrate Levels (CBB)

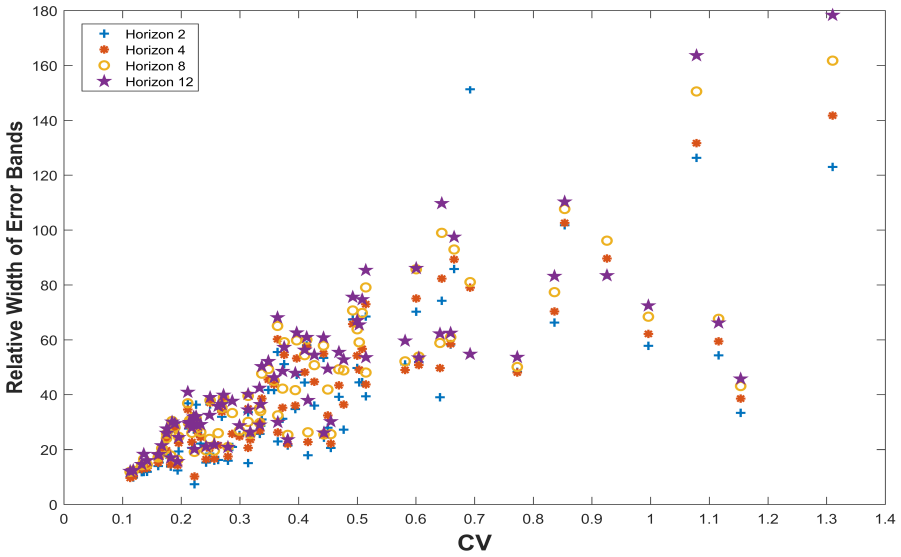
5.15. This shows that the behaviour of CBB varies depending on the network conditions. Furthermore, the forecasting horizon has another impact on the width of the confidence bands (i.e. bitrate levels); however, its impact is also related to the network conditions. Nevertheless, the collected results provide an intuition about the network conditions in which CBB is most efficient and the configurational parameters with which it should be deployed. In more detail, Figure 5.16a provides a closer look at the impact of throughput variation on the width of the confidence interval for the four investigated horizons (i.e. 2, 4, 8, 12). It can be observed from Figure 5.16a that the CBB approach is effective only when the network throughput variation (i.e. CV index) has a value between (0.1–0.3). However, under a highly variable network throughput (i.e. a CV value more than 0.3), the bitrate guidance of CBB is not very practical, as the bounding scheme filters a minimal number of bitrate levels.

5.6.3 Mean Width of Error Bands

Figure 5.17 illustrates the bands' width of the other proposed bounding scheme (i.e. EBB). Here, the bitrate boundaries are computed by duplicating the collected



(a) Relative Conf. Width Versus Throughput CV



(b) Relative Error Width Versus Throughput CV

Figure 5.16

Relative Bands Width Versus Throughput CV

forecasting error. It can be observed from Figure 5.17 that the mean width of the error bands varies depending on the network conditions (i.e. network traces) and configuration parameters of the forecasting algorithm. For example, traces 85, 63, and 91 have the lowest values of the EBB bands, with values of 0.19, 0.22, and 0.32 Mbps respectively; moreover, traces 81, 80, and 78 have the highest error bands, with values of 5.4, 5.46, and 6.12 Mbps respectively. To facilitate understanding of

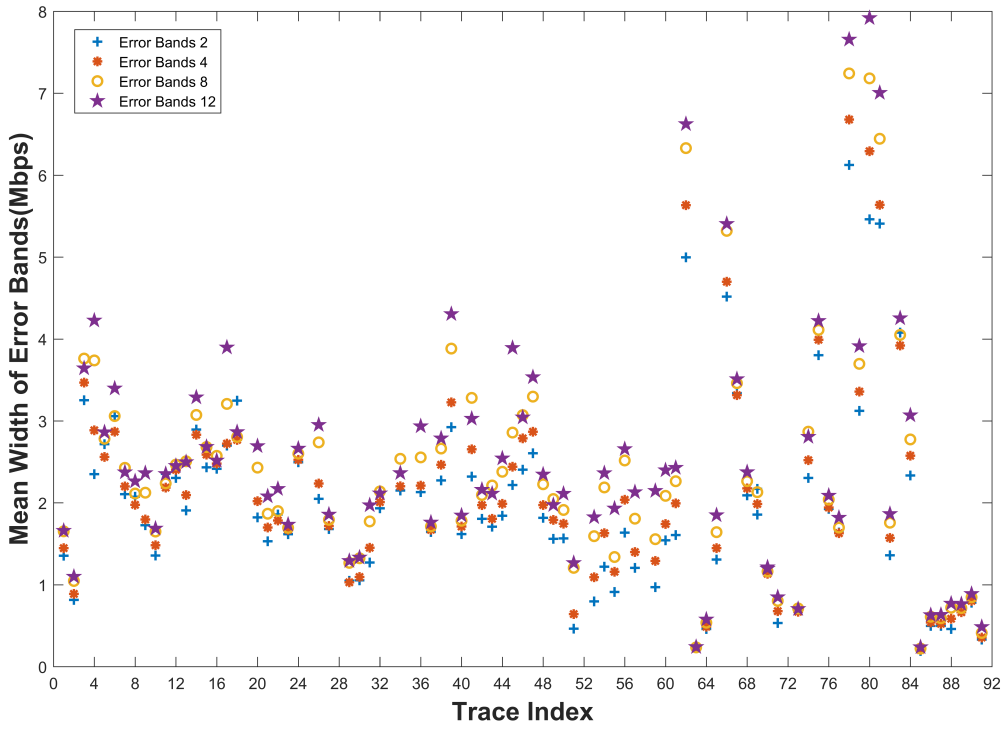


Figure 5.17

Per-trace Mean Width of the Error Bands

the impact of the different network features on the width of the error band, Figure 5.11 presents the degree of correlation between each of the network features plotted against the width of the error bands. It can be seen from Figure 5.11 that CV and skewness are the features most positively correlated with the error bands width, with values of 0.8 and 0.75 respectively. It can therefore also be seen that EBB comes with larger bands width when the throughput level has a high degree of variation. However, the impact of throughput variation (i.e. CV) on the width of the error bands is less than its impact on the Conf. bands. Figure 5.16b presents a closer look at the effects of CV on the relative width of the error bands for the four investigated horizons (i.e. 2, 4, 8, 12). It can be concluded that EBB is able to provide efficient bitrate guidance even with the highly variable network conditions that have a CV level between 0.1–0.8.

Furthermore, Figure 5.18 provides a comparison in term of the bands width between the EBB and the CBB. It can be observed from the figure that the mean width of the error bands is less than the mean width of the confidence bands, as

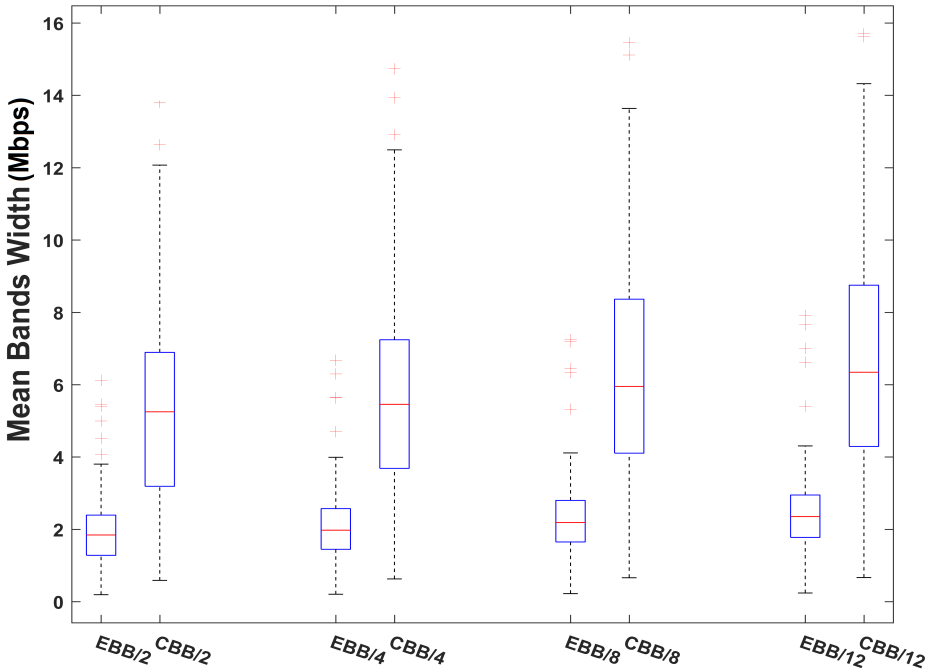


Figure 5.18

Mean Bands Width Comparison

the last bounds the true values with a probability of 0.95, while the former creates boundaries based on the last forecasting error. For example, at the median width of

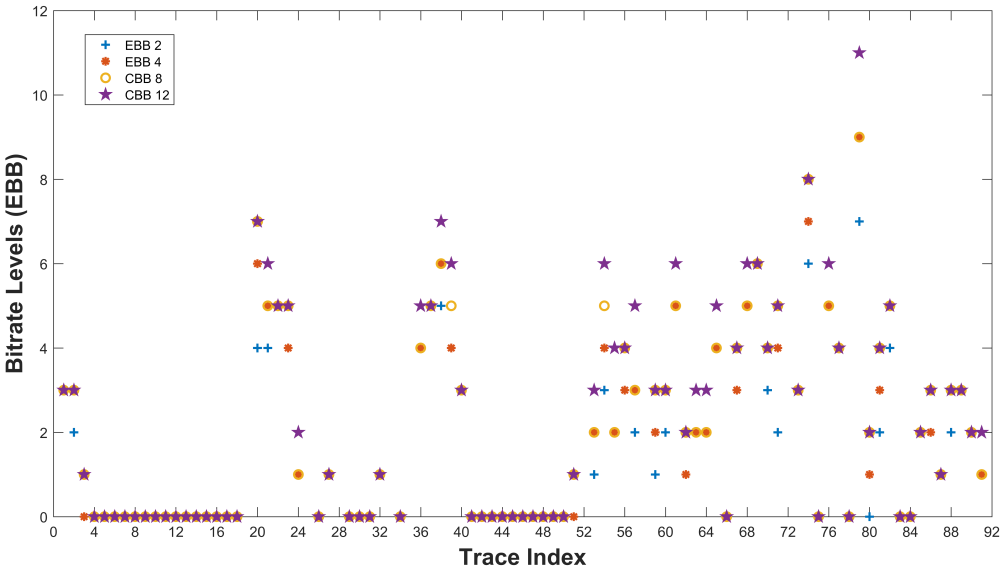


Figure 5.19

Per-trace Number of Bitrate Levels (EBB)

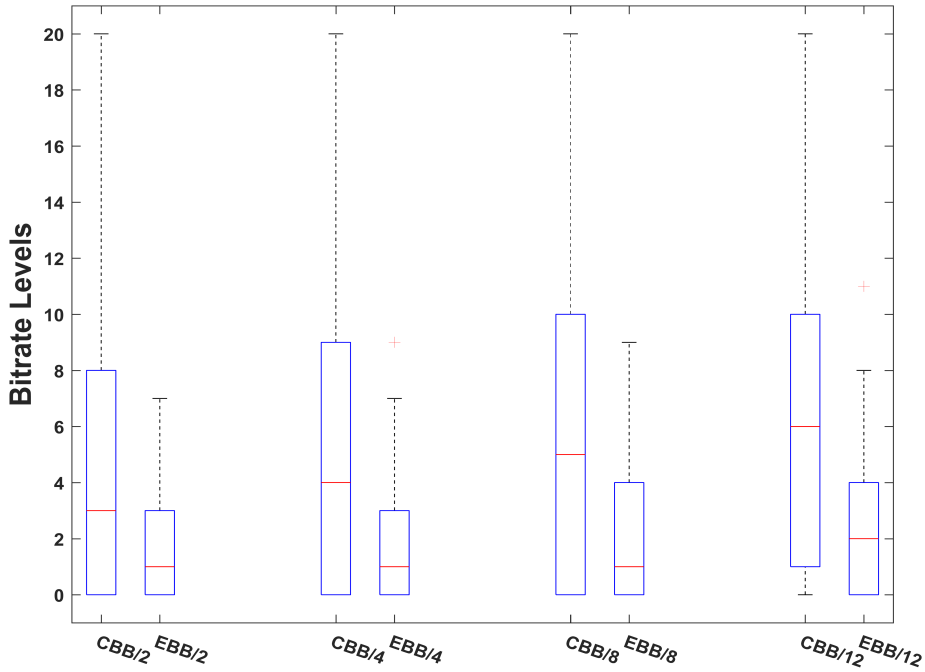


Figure 5.20

Bitrate Level Comparison

EEB, the bands are equal to 1.8 Mbps for horizon 2, but equivalent to 5.2 Mbps with CBB under the same horizon. Furthermore, to understand the impact of the EBB on the bitrate guidance, Figure 5.19 has been constructed to plot the mean number of bitrate levels that EBB offers under each of the examined network traces. It is clear that the mean number of bitrate levels varies depending on the network conditions (i.e. network trace) and configuration parameters of the forecasting algorithm. It is also important to mention that some bitrate levels of some traces are equal to zero, and that this is because the computed minimum and maximum bitrate levels are outside of the maximum available bitrate level of the requested video.

Nevertheless, Figure 5.20 compares the different bounding schemes (i.e. EBB and CBB) in terms of the bitrate levels they offer. It can be seen from the figure that EBB provides lower bitrate levels than CBB across all the investigated horizons, whereas EBB achieves mean bitrate level numbers of 1, 1, 1, and 2 for the corresponding forecasting horizon 2, 4, 8, and 12. Moreover, the other bitrate bounding scheme (i.e. CBB) provides 3, 4, 5, and 6 as the mean numbers of bitrate levels for the pre-mentioned forecasting horizon.

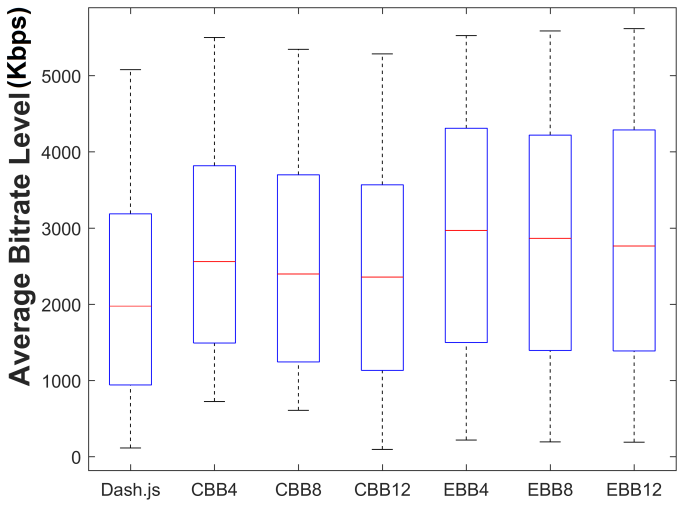


Figure 5.21

Average Bitrate Levels for the Benchmarked Approaches

5.6.4 Bitrate Stability

Figure 5.21 compares the different approaches in terms of the selected video bitrate level when they are deployed under various combinations of the forecasting horizons and network conditions. It is evident from the figure that the EBB approach outperforms the other approaches (i.e. Dash.js and CBB) in term of the bitrate level achieved for all investigated forecasting horizons (i.e. 4, 8 and 12) and under the different network conditions. EBB achieves a median bitrate level of 2.9 Mbps when run under a forecasting horizon of 4, CBB results in a median bitrate level of 2.6 Mbps for the same forecasting, while the achieved bitrate level of dash.js is 2.3 Mbps.

Furthermore, Figure 5.21 also shows that the forecasting horizon of the proposed schemes (i.e. EBB and CBB) has only a mild negative impact on the achieved bitrate levels. For example, EBB achieves mean bitrate levels of 2.9, 2.84, and 2.76 Mbps when run under forecasting horizons of 4, 8, and 12, respectively. Moreover, the enhancement of the achieved bitrate levels when the client is based on the EBB guidance approach is due to the fact that EBB provides a lower number of bitrate levels (i.e. a number closer to the actual throughput level) than CBB does. This, in turn, reduces the likelihood of errors at the client side to stream with very low bitrate levels from the actual available bandwidth and forces the player to stream

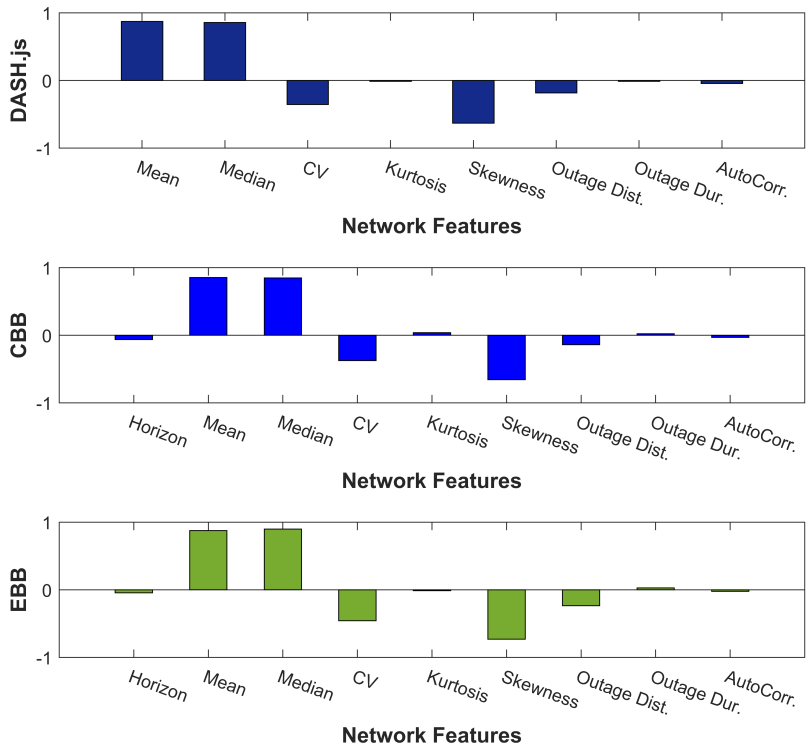


Figure 5.22

Correlation Between Network Features and Selected Bitrate Levels

with the suggested bitrate levels. As shown in Figure 5.21 the horizon of 4 has the highest bitrate levels because it has the highest accuracy; the width of the bands is then inversely proportional to the prediction horizon.

Furthermore, Figure 5.22 illustrates the impact of the network conditions on received bitrate levels for the different approaches. It can be seen from the figure that the different statistical network metrics have different effects on the received bitrate level. Among these different network conditions, the median and mean of throughput have the highest level of correlation with the received bitrate levels at the end client for the various schemes. The skewness level of the network traffic, which represents the lack of symmetry of the given distribution, has a different impact on the received bitrate level, with values of -0.63, -0.65, and -0.72 for dash.js, CBB, and EBB respectively. Throughput variation, which is represented by the CV metric, has the third significant impact on the bitrate level, with values of -0.35, -0.37, and -0.45 for dash.js, CBB, and EBB respectively. These findings are consistent with intuition, as CV and skewness are strongly correlated with the forecasting accuracy,

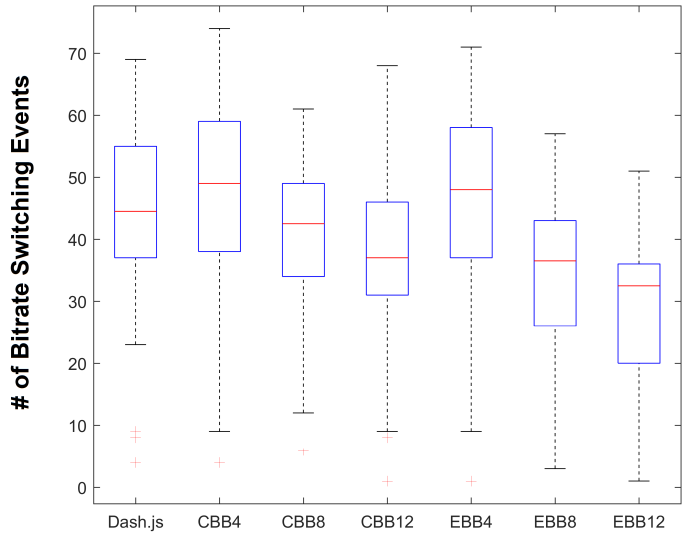


Figure 5.23

Mean Number of Bitrate Switching

and consequently, the number of the bitrate levels. The increase of these metrics therefore leads to an increase in the likelihood of errors at the client side to stream with very low bitrate levels from the actual available bandwidth. Figure 5.22 also presents the impact of the throughput outage on the received bitrate levels; however, its impact is lower than the above-mentioned network features.

Figure 5.23 compares the different approaches in term of bitrate stability, which is represented as bitrate switching (i.e. the number of bitrate switching events). Figure 5.23 indicates that the two approaches (i.e. EBB and CBB) behave similarly, in the sense that the bitrate switching and occurrence of this bitrate switching is proportional to the guidance interval (i.e. forecasting horizon). This is because of the short-forecasting horizon results with lower forecasting error and confidence interval bands, which in turn provide DASH players with very few numbers regarding the bitrate levels at every guidance interval. Therefore, it is more likely to receive a new set of bitrate levels every guidance.

On the other hand, long-term forecasting tends to lead to high forecasting error, which in turn generates a higher number of bitrate levels that may include the current bitrate with which the client is streaming. Therefore, if the client has a sufficient buffer level, it may stay with the previous bitrate level. Despite the fact that EBB has the highest bitrate switching number, it achieves a low bitrate switching

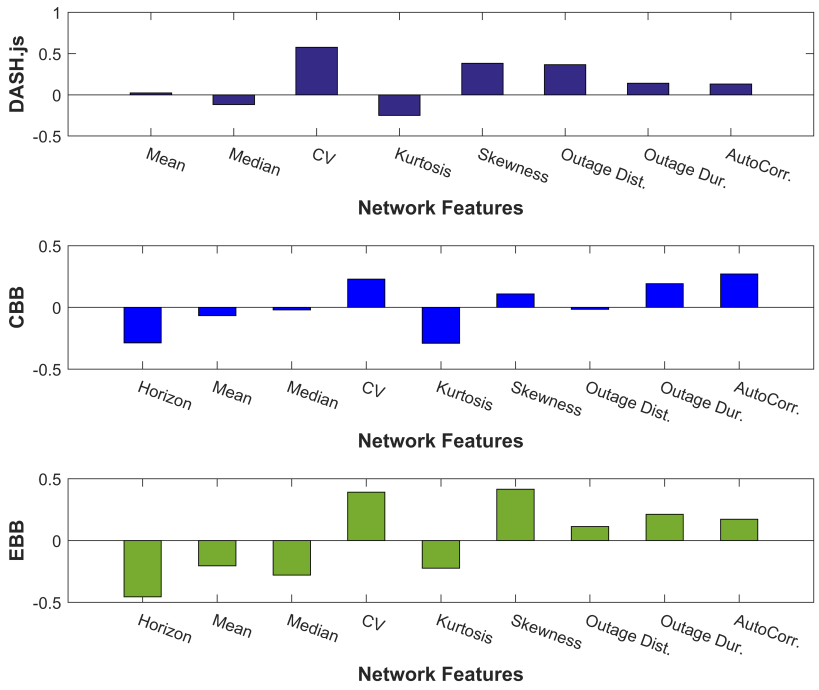


Figure 5.24

Correlation Between Network Features and Number of Bitrate Switching

amplitude value. Figure5.20 presents the number of bitrate levels provided to each player to stream with. As a result, when bitrate switching occurs, the client cannot switch to a bitrate that is lower or higher than the offered boundaries.

Furthermore, Figure 5.24 plots the impact of the network features and forecasting parameters on the bitrate switching. It can be observed that the impact of the forecasting horizon is higher than the impact of network features for the two approaches with values of -0.3 and -0.45 for CBB and EBB, respectively. Throughput variation (i.e. CV) has the highest correlation among the network features with the stability of the received bitrate level, with values of 0.57, 0.22, and 0.4 for dash.js, CBB, and EBB, respectively. Furthermore, skewness has the second highest impact on the bitrate stability for the EBB scheme with a value of 0.39, while its impacts are lower on the other approaches (dash.js and CBB) with values of 0.35 and 0.1 respectively. Time series predictability is represented by the two main network features (Kertusis and Auto-correlation at lag 1), which have the third most significant impact on bitrate stability.

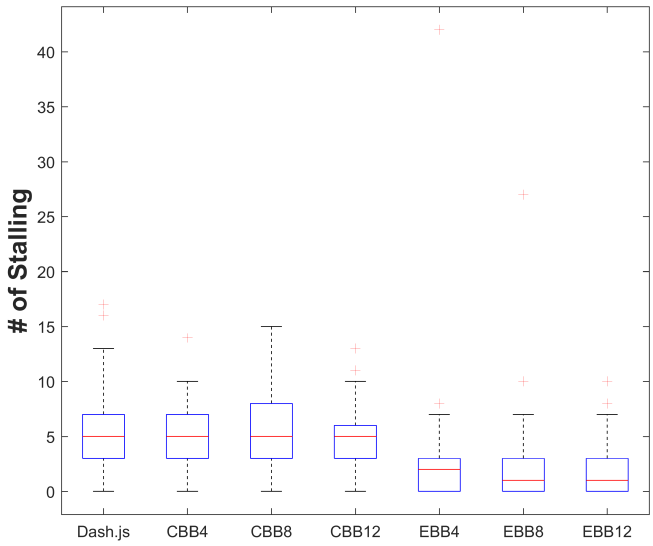


Figure 5.25

Mean Number of Video Stalling Events

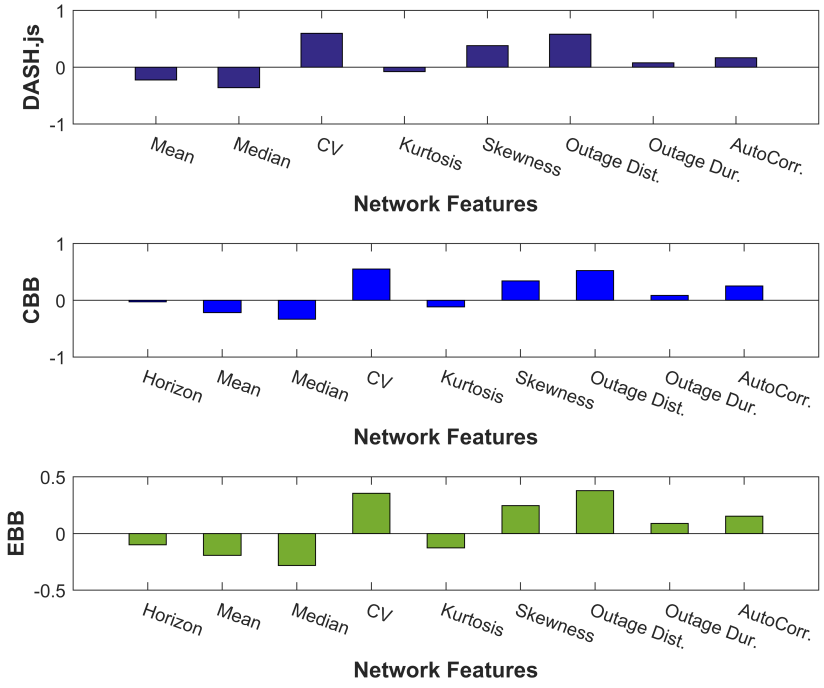


Figure 5.26

Correlation Between Network Features and Number of Video Stalls

5.6.5 The Number of Video Stalling Events and their Duration

Figure 5.25 depicts the number of video stalling events for the investigated approaches under the different network conditions and forecasting horizons. These results indicate that the EBB guidance approach outperforms others in terms of the number of video stalls. The CBB guidance scheme performs similarly to the native ABR algorithms (i.e. dash.js). The performance of EEB with respect to the frequency of video stalling events comes from the fact that, with EBB guidance, DASH clients receive a small number of bitrate levels that are close to the actual network throughput. This, in turn, reduces the possibility of streaming with an outlier bitrate level.

Figure 5.26 demonstrates the impact of the network features on video stalling events for the approaches under investigation. It can readily be seen that throughput variation and throughput outage have the greatest impact on the number of video stalls. There are two main reasons behind the strong correlation between these metrics (i.e. CV and outage) and video stalling events. The first of these relates to the fact that a high throughput CV results in high forecasting errors, which in turn

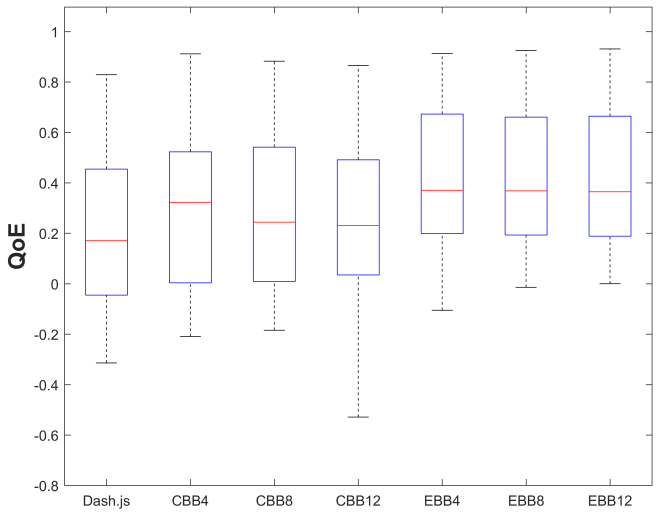


Figure 5.27
Average nQoE for the Benchmarked Approaches

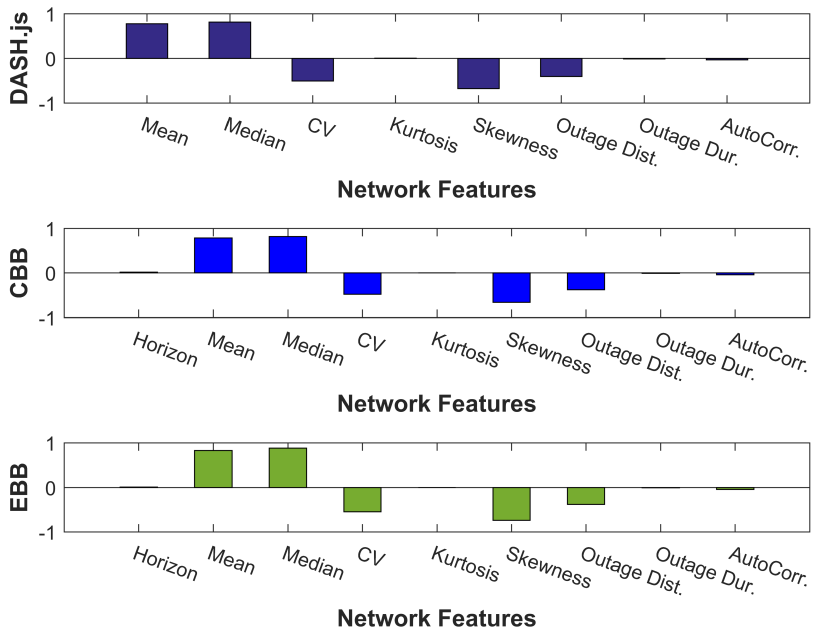


Figure 5.28

Correlation Between Network Features and Received QoE

increase the band width of the bounding schemes (i.e. CBB and EBB). Moreover, misestimating the resource availability that could occur when the player streams under unstable network conditions (i.e. high CV level), increases the number of video stall events. Second, throughput outage results in video buffer underflow; therefore, in order to mitigate the impact of throughput outage, a customised buffer filling strategy must be considered. However, the compared approaches have been investigated with the default buffer filling strategy in use. Other network features, such as autocorrelation at lag 1 and kurtosis, are also correlated with the occurrence of the video stall events. Nevertheless, Figure 5.26 shows that the impact of the forecasting parameters (i.e. horizon) on the video stalling events is lower than the impact of the network features. This is because the latter metric (i.e. forecasting horizon) has a minimal impact on the forecasting accuracy and therefore the number of the filtered bitrate levels.

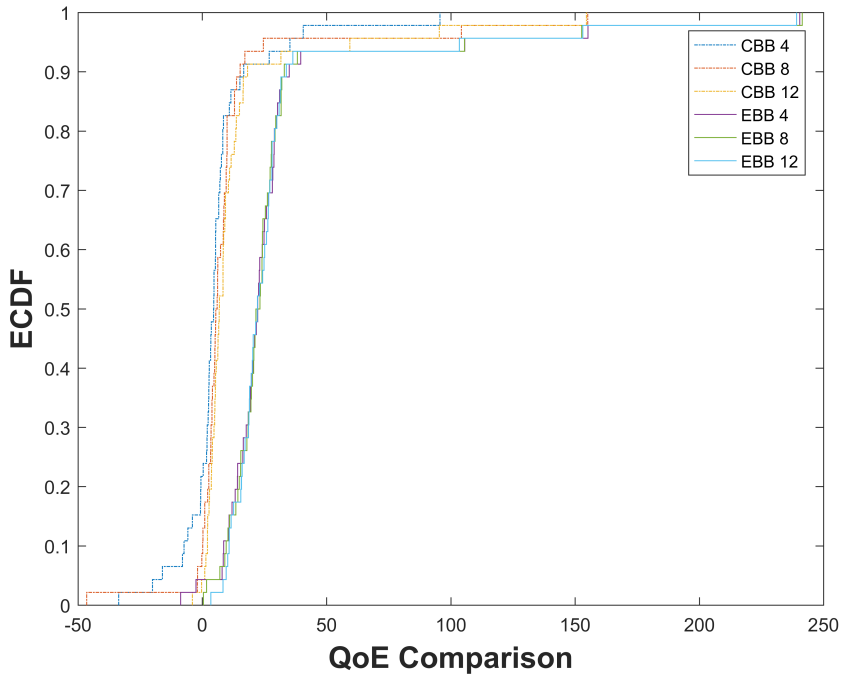


Figure 5.29

QoE Improvement Relative to the Baseline Approach

5.6.6 The Performance of Normalised QoE

Figure 5.27 compares the different approaches in term of normalised QoE when they are deployed under various combinations of the forecasting horizons and network conditions. This figure shows that deploying EBB guidance provides the highest QoE for the end users when compared with other approaches. The reason behind improving the perceived QoE under the deployment of EBB is that the latter provides the highest average bitrate levels, the lowest switching amplitude, and the lowest number of video stalling events. It is worth mentioning here that EBB is more sensitive to prediction accuracy than other approaches. Furthermore, deploying CBB results in a lower perceived QoE at the end client than EBB; this is because CBB provides the clients with a wider range of bitrate levels, which in turn increases the possibility of streaming with an outlier bitrate level.

In addition, Figure 5.28 illustrates the impact of the network conditions on the perceived QoE for the different approaches. It can be noticed that the different network features have different impacts on the perceived QoE. Among these different network features, the median and mean of the throughput have the highest level of

correlation with the perceived QoE at the end client for the various schemes. This is because a high level of network throughput leads the client to stream with a high bitrate level. On the other hand, other network features, such as CV, skewness, and outage, have a negative correlation with the perceived QoE, as these features are proportional to the bitrate switching and video stalling events (i.e. sections 5.6.4 and 5.6.5), which have a negative impact on the perceived QoE.

Nevertheless, from the previous results, it can be concluded that it is quite significant to tune the prediction scheme configuration parameters based on the sensitivity of the QoE model and the network conditions. For example, long-term forecasting is recommended when the QoE model has more weight when compared with bitrate switching, as forecasting horizon is negatively correlated with the video switching events, as depicted in 5.6.4. On the other hand, deploying a bitrate adaptation scheme with a lower horizon could result in a higher QoE when the QoE model is more sensitive toward the achieved bitrate and video stalling.

Finally, Figure 5.29 compares the different approaches in terms of QoE improvement relative to the baseline approach (i.e. dash.js). As shown in the figure, about 60% of the experiments conducted boost the perceived QoE by more than 20% when the EEB guidance scheme is deployed. On the other hand, the CBB bitrate guidance approach results in a 10% improvement in perceived QoE for 60% of the conducted experiments compared with the baseline approach.

5.7 Summary

This chapter has presented a novel QoE-driven network-assisted architecture for HTTP adaptive video streaming, called `BBGDASH`$^{+}$, which optimises the video delivery services by providing each player with the optimal bitrate levels that matches the requirements of the end user and network conditions. To accomplish this, two time-series forecasting approaches – namely, the Error-Based Bounding (EBB) and Confidence-Based Bounding (CBB) algorithms – are proposed to identify the optimal bitrate levels for each client. Furthermore, in order to examine the applicability of the proposed approaches in the wild, extensive evaluations have been conducted to investigate the feasibility of deploying these approaches under different network

conditions. Based on the analysis conducted in this chapter, the following findings and conclusions were drawn: 1) The CBB approach can be effective only when the network throughput variation (i.e. the CV index) has a value between 0.1–0.3; 2) EBB can provide efficient bitrate guidance even under highly variable network conditions that have a CV level between 0.1–0.8; 3) EBB provides a lower number of bitrate levels (i.e. closer to the actual throughput level) than CBB does. This, in turn, reduces the chances of errors at the client side to stream with very low bitrate levels from the actual available bandwidth and force the player to stream with the suggested bitrate levels; 4) Frequent bitrate guidance (i.e. a short forecasting horizon) increases the bitrate switching number for the two bounding approaches; 5) The EBB guidance approach outperforms others in terms of the number of video stalls, while the CBB guidance scheme performs similarly to the native ABR algorithms (i.e. dash.js); 6) Throughput variation (CV) and throughput outage have the highest impact on the number of video stalls; 7) In terms of QoE, EBB bitrate guidance boosts the perceived QoE by 20% relative to the baseline approach (i.e. dash.js), while CBB improves the end users' QoE by 10% compared with the baseline approach.

Chapter 6

Conclusion

6.1 Introduction

This chapter summarises the achievements of this thesis and the solutions that have been presented to address the issues of interest. This chapter also provides the reader with the future directions of the present research and the plan for subsequent works. The structure of this chapter is organised as follows. Section 1 presents the objectives that have been presented in this thesis. Section 2 summarises the evaluation of the conducted experiments for the proposed solutions. Finally, the future directions and the plan for subsequent works are presented in Section 3.

6.2 Contributions

The main contributions introduced within this thesis are presented below.

- A dynamic programming-based algorithm is introduced, based on the concept of Max-Min fairness, to provide QoE-level fairness for the competing HAS players and efficient resource allocation for the available network resources.
- The proposed algorithm is implemented on top of an SDN-based architecture, referred to as BBGDASH. BBGDASH leverages the flexibility of the SDN management and control to deploy the outcome of the proposed algorithm on the application and network levels.
- A novel bitrate guidance approach that provides the guidance for each client

by moving the entire control into an external entity, without relying solely on the pure client decision. To achieve this, the guidance scheme provides the optimal boundary for each client to adapt the requested bitrate levels within the boundaries offered.

- Two time series-based forecasting approaches, Error-Based Boundary (EBB) and Confidence-Based Boundary (CBB), which identify the optimal set of bitrate levels for DASH client base on the network conditions, are also introduced.

- The proposed time series-based algorithms are implemented on top of BBGDASH+, which is an intelligent streaming architecture that extends the architecture of BBGDASH to provide an intelligent guidance under variable network conditions.

6.3 Evaluation

In order to evaluate the proposed algorithms, real frameworks (i.e. BBGDASH and `BBGDASH`$^+$)) have been implemented. The architecture of each of these frameworks consists of a set of open-source components that all collaborate with each other in order to achieve the main objectives. BBGDASH is deployed in order to examine the proposed algorithms under stable network conditions (i.e. wired network conditions), while `BBGDASH`$^+$)) extended the latter former framework to evaluate the algorithms under wireless network conditions. Within the evaluation, the most important QoE-based metrics have been considered: these include video bitrate level, video switching events, video stalling events, normalised QoE, fairness level, and network resource underutilisation.

The findings can be summarised as follows:

- Most of the purely client-based ABR algorithms are not efficient enough to provide stable video delivery, as they compete to access a common network bottleneck.

- Under wireless network conditions, existing purely client-based ABR algorithms fail to utilise the available network resources and thus stream at a

lower bitrate level than the available resources.

- The experimental evaluation of a video streaming scenario, with nine heterogeneous DASH players using a proof-of-concept implementation, demonstrated the potential of the proposed approach when compared with the other benchmarked solutions. In this specific scenario, BBGDASH outperforms the other solutions and increases network resource utilisation by up to 36%, and QoE up to 38%, relative to the other comparison approaches.

- `BBGDASH`$^{+}$ optimises the video delivery services by providing each player with optimal bitrate levels, which match the requirements of the end user and network conditions. In more detail, the following findings can summarise the performance of `BBGDASH`$^{+}$: 1) The CBB approach can be effective only when the network throughput variation (i.e. the CV index) has a value between 0.1–0.3; 2) EBB can provide efficient bitrate guidance even under highly variable network conditions that have a CV level between 0.1–0.8; 3) EBB provides a lower number of bitrate levels (i.e. closer to the actual throughput level) than CBB does. This, in turn, reduces the chances of errors at the client side to stream with very low bitrate levels from the actual available bandwidth and force the player to stream with the suggested bitrate levels; 4) Frequent bitrate guidance (i.e. a short forecasting horizon) increases the bitrate switching number for the two bounding approaches; 5) The EBB guidance approach outperforms others in terms of the number of video stalls, while the CBB guidance scheme performs similarly to the native ABR algorithms (i.e. dash.js); 6) Throughput variation (CV) and throughput outage have the highest impact on the number of video stalls; 7) In terms of QoE, EBB bitrate guidance boosts the perceived QoE by 20% relative to the baseline approach (i.e. dash.js), while CBB improves the end users' QoE by 10% compared with the baseline approach.

6.4 Future Work

Video streaming services take two main forms: live video streaming and video-on-demand streaming (VoD). Within this thesis, the bounding bitrate guidance

approach has been investigated in depth under VoD scenarios. However, the other form of video streaming services (i.e. live video streaming) is rapidly increasing in popularity, which imposes an additional burden on live video service architectures such as YouTube Live and makes it challenging to deliver video with a high QoE to the end user. It is therefore necessary to present novel solutions that improve the performance of live video streaming services. Bounding bitrate guidance has shown its efficiency at providing stable video delivery with minimal cost and communication between the network-based component and the DASH client. Therefore, it is worth investigating the applicability of the bounding bitrate guidance on the top of live video delivery. In the live streaming scenario, the latency in video delivery has the highest impact on end-user QoE. Furthermore, video stalling events that occur when the client streams at a higher bitrate level than the available network bandwidth have a negative influence on video latency. It is accordingly expected that deploying the bounded bitrate guidance within the live streaming scenario brings significant benefits to QoE metrics and QoE overall.

Another potential avenue for extending this thesis involves applying the bounding approach to buffer management. The bounding guidance approach has been presented within this thesis in an attempt to identify the set of bitrate levels used by each DASH player to adapt the requested segment. However, buffer level has a significant impact on the stability of the ABR algorithms. It is thus very interesting to apply the bounding guidance approach to the identification of the minimum and maximum buffer size based on the user context and the type of streaming service.

Furthermore, BBGDASH and `BBGDASH`$^{+}$ consider the presence of the collaboration between the network operator and the service provider. In order to implement this collaboration, the Server and Network Assisted DASH (SAND) was implemented within these frameworks. However, the collaboration between the network operator and the service provider is not always present; therefore, these solutions need to be extended so that they are applicable even when such collaboration is absent. To achieve this, QoE metrics need to be detected by the network operator based on network traffic. Different techniques, such as deep packet inspection (DPI), are used by the network operators to detect QoE metrics. However, a large portion of the Internet traffic is encrypted [134], which makes such techniques inapplicable

to the estimation of QoE metrics. There is thus a need for alternative techniques for estimating the QoE level of the encrypted HAS/DASH traffic. Machine learning can play a crucial role in these cases, which is why a large body of research has been focused on this area [11][12].

In addition, the presented frameworks can be extended to manage not only the requested bitrate level, but also user preferences for the multi-access network architecture. MPTCP was recently presented by the Internet Engineering Task Force (IETF) as a multipath protocol that enables sending and receiving data transparently and simultaneously over the multiple paths. However, MPTCP lacks the flexibility to prioritise one link over another and support for path control. Hence, video streaming over MPTCP may over-utilise the available network resources, such as metered cellular links.

Han et al. [65] also provided a set of measurements for the throughput and RTT of WiFi networks throughout 33 different locations in the US states, conducted in indoor and outdoor environments. The results of this study showed that 64% of the WiFi assessed during these experiments is not efficient enough to support the highest bitrate of a 1080p video; 15% of this WiFi can sometimes support the highest quality, while 20% of these devices almost support the highest quality. This result indicates that MPTCP can improve the received QoE for video streaming. However, this approach may over-consume other resources, such as cellular data usage. For example, the throughput provided by WiFi and LTE are 8 and 9 Mbps respectively, while the highest bitrate of the requested 1080p video is 4 Mbps; however, under MPTCP, half of or more of the requested data is sent over the LTE, while even WiFi can support the requested bitrate. To this end, video streaming over MPTCP must be controlled in order to dynamically control MPTCP paths and improve the QoE received by the end-user.

The different solutions that have been presented within this thesis are aimed at maximising the perceived QoE for the average user. However, in order to provide efficient utilisation of the available resources and a good experience for the end user, it is essential to consider the user's context. For example, many users leave the streaming sessions, or shift their attention to other activities (such as listening to YouTube or editing a document). In such scenarios, inactive (passive) users

consume more bandwidth than required to ensure a satisfying QoE; this is because, without user attention, the received video quality becomes irrelevant. This could lead to excessive resource and power consumption; moreover, it could also influence the QoE received by the active users in cases of limited bandwidth. In parallel, the introduction of SDN opens a path towards better resource utilisation through considering the network and application-level information. With this in place, the greatest possible gain could be achieved if the context information was considered alongside the allocation of the available resources.

References

[1] Adobe http dynamic streaming. http://www.adobe.com/products/hds-dynamic-streaming.html. Accessed: 2019-12-20.

[2] Adobe. Real-Time Messaging Protocol (RTMP),. http://www.adobe.com/devnet/rtmp.html.

[3] Apple http live streaming. https://developer.apple.com/streaming/. Accessed: 2019-12-20.

[4] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper ,. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html.

[5] DASH External Dataset. http://www-itec.uni-klu.ac. at/ftp/datasets/DASH-Dataset2014/.

[6] Dash industry forum. https://dashif.org/. Accessed: 2019-12-20.

[7] DASH Industry Forum. http://www.dashif.org/.

[8] Ffmpeg. https://www.ffmpeg.org/. Acce019-12-20.

[9] GPAC:Multimedia Open Source Project,. [Availableat]:https://gpac.wp.imt.fr/mp4box/.

[10] GPAC:Multimedia Open Source Project. *What's New*, pages 0–60.

[11] ISO/IEC 23009-1:2014 - Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats.

[12] Microsoft silverlight smooth streaming. https://www.microsoft.com/silverlight/smoothstreaming. Accessed: 2020-03-03.

[13] Microsoft smooth streaming. http://www.iis.net/downloads/microsoft/smooth-streaming. Accessed: 2019-12-20.

[14] Mininet: An instant virtual network. http://mininet.org/. Acce019-12-20.

[15] Openflow switch specification. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf/. Accessed: 2019-12-20.

[16] Q1 2017 State of the Internet - Connectivity Report — Akamai.

[17] RYU SDN Framework. http://osrg.github.io/ryu/.

[18] Video quality experts group (vqeg). https://www.its.bldrdoc.gov/vqeg/vqeg-home.aspx/. Accessed: 2019-12-20.

[19] ITU-T Rec. P.910: Subjective video quality assessment methods for multimedia applications, April 2008.

[20] ITU-T Rec. P.10/G.100: Vocabulary for performance and quality of service. Amendment 5: New definitions for inclusion in Recommendation ITU-T P.10/G.100, July 2016.

[21] ISO/IEC 23009-5:2017. Information Technology – Dynamic Adaptive Streaming over HTTP (DASH) – Part 5: Server and Network Assisted DASH (SAND). Standard. International Organization for Standardization. 2017.

[22] R. Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[23] I. Ahmed and A. Petlund. Analysing user satisfaction in next generation networks for multimedia multicast transmission. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 5 2015.

[24] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth. *Proc. ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'12)*, page 6, 2012.

[25] S. Akhshabi, A. C. Begen, and C. Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. *Methodology*, pages 157–168, 2011.

[26] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang. Oboe. *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '18*, 1:44–58, 2018.

[27] A. E. Al-Issa, A. Bentaleb, A. A. Barakabitze, T. Zinner, and B. Ghita. Bandwidth Prediction Schemes for Defining Bitrate Levels in SDN-enabled Adaptive Streaming. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–7, 2019.

[28] A. E. Al-Issa, A. Bentaleb, T. Zinner, I. Mkwawa, and B. Ghita. BBG-DASH: A Max-Min Bounded Bitrate Guidance for SDN Enabled Adaptive Video Streaming. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 307–314, 2019.

[29] A. Al-Jawad, P. Shah, O. Gemikonakli, and R. Trestian. LearnQoS: A learning approach for optimizing QoS over multimedia-based SDNs. *in Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast (BMSB)*, pages 1–6, June 2018.

[30] T. Bakhshi and B. Ghita. User traffic profiling. In *2015 Internet Technologies and Applications (ITA)*, pages 91–97. IEEE, 2015.

[31] T. Bakhshi and B. Ghita. User-centric traffic optimization in residential software defined networks. In *2016 23rd International conference on telecommunications (ICT)*, pages 1–6. IEEE, 2016.

[32] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a predictive model of quality of experience for internet video. *In ACM SIGCOMM*, pages 339–350, August 2013.

[33] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori. Qoe management of multimedia streaming services in future networks: a tutorial and survey. *IEEE Communications Surveys & Tutorials*, 22(1):526–565, 2019.

[34] A. Beben, P. Wiśniewski, J. M. Batalla, and P. Krawiec. Abma+: lightweight and efficient algorithm for http adaptive streaming. In *Proceedings of the 7th International Conference on Multimedia Systems*, page 2. ACM, 2016.

[35] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann. Want to play DASH? A Game Theoretic Approach for Adaptive Streaming over HTTP. *Proceedings of the 9th ACM Multimedia Systems Conference on - MMSys '18*, pages 13–26, 2018.

[36] A. Bentaleb, A. C. Begen, and R. Zimmermann. SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1296–1305, 2016.

[37] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous. SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming. *IEEE Transactions on Multimedia*, 19(10):2136–2151, 2017.

[38] A. Bentaleb, C. Timmerer, A. C. Begen, and R. Zimmermann. Bandwidth prediction in low-latency chunked streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 7–13. ACM, 2019.

[39] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz. Network assisted content distribution for adaptive bitrate video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 62–75. ACM, 2017.

[40] G. Bjontegaard. Calculation of average PSNR differences between RD-curves (VCEG-M33). ITU-T Video Coding Experts Group (April 2001).

[41] F. Bossen. Common Test Conditions and Software Reference Configurations for HM (JCTVC-L1100). Jan 2013. ITU-T and ISO/IEC JTC 1 Joint Collaborative Team on Video Coding.

[42] N. Bouten, M. Claeys, B. V. Poecke, S. Latre, and F. D. Turck. Dynamic Server Selection Strategy for Multi-Server HTTP Adaptive Streaming Services. *12th International Conference on Network and Service Management (CNSM)*, pages 82–90, 2016.

[43] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, et al. Qualinet white paper on definitions of quality of experience. 2013.

[44] L. C, B. Imed, and G. Moncef. Rate Adaptation for Adaptive HTTP Streaming. *Proceedings of the second annual ACM conference on Multimedia systems*, 22(1):169–174, February 2011.

[45] M. S. C. Cetinkaya, E. Karayer and C. Hellge. SDN for Segment Based Flow Routing of DASH. *Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin)*, pages 74–77, 2014.

[46] P. L. Callet, S. Möller, and A. Perkis. Qualinet White Paper on Definitions of Quality of Experience (2012). *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, 1.2, March 2012.

[47] G. Calvigioni, R. Aparicio-Pardo, L. Sassatelli, J. Leguay, P. Medagliani, and S. Paris. Quality of experience-based routing of video traffic for overlay and isp networks. *IEEE International Conference on Computer Communications (INFOCOM)*, pages 121–132, 2018.

[48] G. Cermak, M. Pinson, and S. Wolf. The relationship among video quality, screen resolution, and bit rate. *IEEE Transactions on Broadcasting*, 57(2):258–262, 2011.

[49] C. Chatfield. *The analysis of time series: an introduction*. CRC Press, Florida, US, 6th edition, 2004.

[50] F. Chiariotti, L. T. Stefano DAronco, and P. Frossard. Online learning adaptation strategy for dash clients. *Proceedings of the 7th International Conference on Multimedia Systems*, 18:738–751, 2016.

[51] G. Cofano, L. D. Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2017.

[52] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon. Cross-layer scheduler for video streaming over MPTCP,. *Proceedings of the 7th International Conference on Multimedia Systems,*, May 2016.

[53] P. J. Diggle. Time series; a biostatistical introduction. Technical report, 1990.

[54] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. *In ACM SIGCOMM*, pages 362–373, August 2011.

[55] O. Dobrijevic, M. Santl, and M. Matijasevic. Ant colony optimization for QoE-centric flow routing in software-defined networks. *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*, pages 274–278, 2015.

[56] H. E. Egilmez, S. Civanlar, and A. M. Tekalp. An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks. *IEEE Transactions on Multimedia*, 15(3):710–715, April 2013.

[57] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella. D-dash: A deep q-learning framework for dash video streaming. *IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING*, 3(4):703–718, 2017.

[58] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. *ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking, ser. FhMN*, pages 120–123, December 2013.

[59] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies - CoNEXT '16*, pages 129–143, 2016.

[60] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. Quantification of YouTube QoE via crowdsourcing. *Proceedings - 2011 IEEE InternationalSymposium on Multimedia, ISM 2011*, pages 494–499, 2011.

[61] T. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. *In ACM IMC*, pages 225–238, November 2012.

[62] T. Huang, X. Yao, C. Wu, R.-X. Zhang, and L. Sun. Tiyuntsong: A self-play reinforcement learning approach for abr video streaming. *arXiv:1811.06166*, May 2019.

[63] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun. Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. *Proceedings of the 26th ACM international conference on Multimedia*, October 2018.

[64] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. Using the Buffer to Avoid Rebuffers: Evidence from a Large Video Streaming Service. *Proc. ACM Conf. SIGCOMM,*, pages 187–198, 2016.

[65] X. Huang, T. Yuan, G. Qiao, and Y. Ren. Deep reinforcement learning for multimedia traffic control in software defined networking. *Computer Networks*, 32(6):35–41, 2018.

[66] R. J. Hyndman, Y. Khandakar, et al. *Automatic time series for forecasting: the forecast package for R.* Number 6/07. Monash University, Department of Econometrics and Business Statistics, 2007.

[67] N. Hyunwoo, D. Calin, and H. Schulzrinne. Towards Dynamic MPTCP Path Control Using SDN. *IEEE NetSoft Conference and Workshops (NetSoft)*, pages 286–294, July 2016.

[68] C. James, E. Halepovic, M. Wang, R. Jana, and N. K. Shankaranarayanan. Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH? *IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS),*, pages 331–336.

[69] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang. {CFA}: A practical prediction system for video qoe optimization. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 137–150, 2016.

[70] 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, 2014.

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

[72] J. Junchen, S. Vyas, and H. Zhang. Improving Fairness, Efficiency and Stability in HTTP-Based Adaptive Video Streaming. *IEEE/ACM Transactions on Networking*, 22(1):326–340, January 2014.

[73] A. Khalid, A. H. Zahran, and C. J. Sreenan. An sdn-based device-aware live video service for inter-domain adaptive bitrate streaming. *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 121–132, 2019.

[74] N. Khambari, B. Ghita, and L. Sun. Qoe-driven video enhancements in wireless networks through predictive packet drops. In *2017 IEEE 13th Interna-*

tional Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 355–361. IEEE, 2017.

[75] A. Khan, L. Sun, and E. Ifeachor. Qoe prediction model and its application in video quality adaptation over umts networks. *IEEE Transactions on Multimedia*, 14(2):431–442, 2011.

[76] N. Khan, M. M. Nasralla, and M. G. Martini. Network and user centric performance analysis of scheduling strategies for video streaming over lte. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1753–1758. IEEE, 2015.

[77] J. W. Kleinrouweler, S. Cabrero, and P. Cesar. Delivering stable high-quality video: An SDN architecture with DASH assisting network elements. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–10, 2016.

[78] J. W. Kleinrouweler, B. Meixner, J. Bosman, H. Van Den Berg, R. Van Der Mei, and P. Cesar. Improving mobile video quality through predictive channel quality based buffering. In *2018 30th International Teletraffic Congress (ITC 30)*, volume 1, pages 236–244. IEEE, 2018.

[79] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[80] J. Kua, G. Armitage, and P. Branch. A survey of rate adaptation techniques for dynamic adaptive streaming over http. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866, 2017.

[81] T. Lan, D. Kao, M. Chiang, and A. Sabharwal. An axiomatic theory of fairness in network resource allocation. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[82] P. Le Callet, C. Viard-Gaudin, and D. Barba. A convolutional neural network approach for objective video quality assessment. *IEEE transactions on neural networks*, 17(5):1316–1327, 2006.

[83] 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, 2014.

[84] E. Liotou, K. Samdanis, E. Pateromichelakis, N. Passas, and L. Merakos. QoE-SDN App: A rate-guided QoE-aware SDN-app for http adaptive video streaming. *IEEE Journal on Selected Areas in Communications*, 36(3):598–615, 2018.

[85] E. Liotou, D. Tsolkas, N. Passas, and L. Merakos. Quality of experience management in mobile cellular networks: key issues and design challenges. *IEEE Communications Magazine*, 53(7):145–153, 2015.

[86] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby. User experience modeling for dash video. In *2013 20th International Packet Video Workshop*, pages 1–8. IEEE, 2013.

[87] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. Deriving and validating user experience model for dash video streaming. *IEEE Transactions on Broadcasting*, 61(4):651–665, Dec 2015.

[88] K. Ma, W. Liu, Z. W. Tongliang Liu, and D. Tao. dipiq: Blind image quality assessment by learning-to-rank discriminable image pairs. *IEEE Transactions on Image Processing*, 26(8):3951–3964, Aug 2017.

[89] R. D. Major and M. B. Hurst. Apparatus, system, and method for adaptive-rate shifting of streaming content, Oct. 21 2014. US Patent 8,868,772.

[90] A. Majumdar, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung. Multicast and Unicast Real-Time Video Streaming Over Wireless LANs. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 12(6):524–534, June 2011.

[91] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018.

[92] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi. Video through a crystal ball: Effect of bandwidth prediction quality on adaptive streaming in mobile environments. In *Proceedings of the 8th International Workshop on Mobile Video*, page 1. ACM, 2016.

[93] A. Mansy, M. Fayed, and M. Ammar. Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming. *IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2015.

[94] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017.

[95] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[96] K. Miller, A.-K. Al-Tamimi, and A. Wolisz. Qoe-based low-delay live streaming using throughput predictions. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 13(1):4, 2017.

[97] X. Min, G. Zhai, K. Gu, Y. Liu, and X. Yang. Blind image quality estimation via distortion aggravation. *IEEE TRANSACTIONS ON BROADCASTING*, 64(2):508–517, March 2018.

[98] A. Mittal, M. A. Saad, and A. C. Bovik. A completely blind video integrity oracle. *IEEE Transactions on Image Processing*, 25(1):289–300, 2015.

[99] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. QDASH: A QoE-aware DASH system. *in Proceedings of the 3rd Multimedia Systems Conference on - MMSys*, 2012.

[100] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. de Veciana. Understanding the impact of video quality on user engagement. *IEEE Journal of Selected Topics in Signal Processing*, pages 652–671, August 2012.

[101] Netflix. VMAF - Video Multi-Method Assessment Fusion. https://github.com/Netflix/vmaf. [Online: Accessed 06-September-2019].

[102] Netflix. Toward A Practical Perceptual Video Quality Metric. https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652, June 2016. [Online: Accessed 06-September-2019].

[103] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[104] O. Oyman and S. Singh. Quality of experience for http adaptive streaming services. *IEEE Communications Magazine*, 50(4):20–27, 2012.

[105] I. M. Ozcelik and C. Ersoy. Chunk duration–aware sdn-assisted dash. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(3):82, 2019.

[106] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[107] S. Petrangeli, J. Famaey, M. Claeys, S. Latre, and F. D. Turck. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl*, pages 1–9, October 2012.

[108] S. Petrangeli, T.Wu, T.Wauters, R. Huysegems, T. Bostoen, and F. D. Turck. A machine learning-based framework for preventing video freezes in HTTP adaptive streaming. *Journal of Network and Computer Applications*, pages 78–92, 2017.

[109] M. H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, Sept 2004.

[110] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue. Abr streaming of vbr-encoded videos: characterization, challenges, and solutions. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, pages 366–378. ACM, 2018.

[111] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello. Incorporating prediction into adaptive streaming algorithms: A qoe perspective. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 49–54, 2018.

[112] I. Rec. J. 144:"objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference". *International Telecommunication Union, Telecommunication standardization sector*, 2004.

[113] A. Rehman, K. Zeng, and Z. Wang. Display device-adapted video quality-of-experience assessment. In *Human Vision and Electronic Imaging XX*, volume 9394, page 939406. International Society for Optics and Photonics, 2015.

[114] U. Reiter, K. Brunnström, K. D. Moor, M.-C. Larabi, M. Pereira, A. Pinheiro, J. You, and A. Zgank. Factors Influencing Quality of Experience: in Quality of Experience: Advanced Concepts, Applications and Methods,. *Springer International Publishing*, pages 55–72, March 2014.

[115] D. Z. Rodriguez, R. L. Rosa, E. C. Alfaia, J. I. Abrahao, and G. Bressan. Video quality metric for streaming service using dash standard. *IEEE Transactions on Broadcasting*, 56(4):628–639, Sept 2016.

[116] D. Z. Rodríguez, Z. Wang, R. L. Rosa, and G. 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), 2014.

[117] M. A. Saad, A. C. Bovik, and C. Charrier. Blind prediction of natural video quality. *IEEE Transactions on Image Processing*, 23(3):1352–1365, 2014.

[118] J. Samain, G. Carofiglio, M. Tortelli, and D. Rossi. A simple yet effective network-assisted signal for enhanced dash quality of experience. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 55–60. ACM, 2018.

[119] M. Sayit, C. Cetinkay, H. Yildiz, and B. Tavli. Dash-qos: A scalable network layer service differentiation architecture for dash over sdn. *Computer Networks*, 154(8):12–25, 2019.

[120] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP),. `https://tools.ietf.org/html/rfc232611`.

[121] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials*, 17(1):469–492, 2015.

[122] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang. Understanding the impact of network dynamics on mobile video user engagement. *In ACM SIGMETRICS*, pages 367–379, June 2014.

[123] Y. Shuai and T. Herfet. On Stabilizing Buffer Dynamics for Adaptive Video Streaming with a Small Buffering Delay. *In ACM NOSSDAV*, Jan 2017.

[124] C. Sieber, W. K. Korbinian Hagn, C. Moldovan, and T. Hoßfeld. Towards machine learning-based optimal has. *arXiv:1808.08065*, Aug 2018.

[125] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen. A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(2s):2 –28, May 2018.

[126] A. Sobhani, A. Yassine, and S. Shirmohammadi. A fuzzy-based rate adaptation controller for dash. *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 31–36, March 2015.

[127] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. *Proc.-IEEE INFOCOM*, 2014.

[128] N. Staelens, S. Moens, W. V. den Broeck, I. Marien, B. Vermeulen, P. Lambert, R. V. de Walle, and P. Demeester. Assessing quality of experience of iptv and video on demand services in real-life environments. *IEEE Transactions on Broadcasting*, 56(4):458–466, Dec 2010.

[129] T. Stockhammer. Dynamic Adaptive Streaming over HTTP–: Standards and Design Principles. *In Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[130] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey. Enhancing MPEG DASH Performance via Server and Network Assistance. *SMPTE Motion Imaging Journal*, 126(1):22–27, 2017.

[131] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, M.-L. Champel, and O. Oyman. Application of sand technology in dash-enabled content delivery networks and server environments. *SMPTE Motion Imaging Journal*, 127(1):48–54, 2018.

[132] C. Timmerer and A. C. Begen. Over-the-top content delivery: State of the art and challenges ahead. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1231–1232, 2014.

[133] Y. Tong, H.Konik, F. A.Cheikh, and A. Tremeau. Full reference image quality assessment based on saliency map analysis. *Journal of Imaging Science and Technology*, 54(3):305031–3050314, 2010.

[134] A. Ulliac and B. V. Ghita. Non-intrusive identification of peer-to-peer traffic. In *2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, pages 116–121. IEEE, 2010.

[135] S. C. V. Jacobson, R. Frederick and H. Schulzrinne. Real-time transport protocol (rtp). `https://www.ietf.org/rfc/rfc3550.txt`. Accessed: 2019-12-20.

[136] M. Venkataraman and M. Chatterjee. Inferring video qoe in real time. *IEEE network*, 25(1):4–13, 2011.

[137] Q. Wang, H.-N. Dai, and D. W. Xiao. Data analysis on video streaming qoe over mobile networks. *EURASIP Journal on Wireless Communications and Networking*, 173:1–3, December 2018.

[138] S. Wang, D. Zheng, J. Zhao, W. J. Tam, and F. Speranza. An image quality evaluation method based on digital watermarking. *IEEE transactions on circuits and systems for video technology*, 17(1):98–105, 2006.

[139] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[140] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen. Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing*, 15(9):2345–2361, March 2016.

[141] P. K. Yadav, A. Shafiei, and W. T. Ooi. Quetra: A queuing theory approach to dash rate adaptation. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1130–1138. ACM, 2017.

[142] F. Y. Yan, H. Ayers, C. Zhu, and S. Fouladi. Continual learning improves internet video streaming. *arXiv:1906.01113*, Sep 2019.

[143] F. Yang, S. Wan, Q. Xie, and H. R. Wu. No-reference quality assessment for networked video via primary analysis of bit stream. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(11):1544–1554, 2010.

[144] S. Yi, Y. Xiaoqi, J. Junchen, S. Vyas, L. Fuyuan, W. Nanshu, L. Tao, and S. Bruno. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. *Proc. Conf. ACM SIGCOMM*, pages 272–285, August 2016.

[145] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *Sigcomm 2015*, pages 325–338, 2015.

[146] L. Yitong, S. Yun, M. Yinian, L. Jing, L. Qi, and Y. Dacheng. A study on quality of experience for adaptive streaming service. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 682–686. IEEE, 2013.

[147] C. Zhou and Z. Lin, C.W andGuo. mdash: A markov decision-based rate adaptation approach for dynamic http streaming. *IEEE Trans. Multimed*, 18:738–751, 2016.

[148] W. Zhou, L. Li, M. Luo, and W. Chou. REST API design patterns for SDN northbound API. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 358–365. IEEE, 2014.

[149] T. Zinner, M. Jarschel, A. Blenk, F.Wamser, and W. Kellerer. Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects and Challenges. *IEEE Network Operations and Management Symposium (NOMS)*, pages 1–6, 2014.

[150] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, , and R. K. Sinha. Can Accurate Predictions Improve Video Streaming in Cellular Networks. *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 45(4):57–62, Feb 2015.