Advanced Modelling of Adaptive Bitrate Selection

Yusuf Sani

School of Computing and Communications

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

2017

Abstract

Nowadays, a typical video content provider serves a variety of platforms e.g. smartphones, web browsers, and smart TVs. Each of these platforms has specific requirements with respect to transmission and video quality. Moreover, since these devices are increasingly being used on-the-go, the environment within which most of these video streaming clients operate is both unreliable and time-varying. To cater for these heterogeneous requirements, content providers are increasingly adopting adaptive streaming services. Through such services, the quality of the video content received by a user is adapted to fit its specific requirements and capabilities.

To adapt the video quality, system capabilities such as network capacity and memory have to be continuously monitored and measured, chunk requests have to be scheduled, and then the optimal video rate has to be decided. Each of these tasks is usually managed by a sub-module of the adaptive bitrate selection function. However, these sub-components interact in a non-trivial manner. For example, while on-off chunk scheduling helps to prevent buffer overflow, it negatively affects the TCP throughput. Hence, these complex interactions between these different sub-components of the adaptive streaming algorithm result in unnecessary rebufferings, undesirable variability, and sub-optimal video quality.

To help simplify these interactions, this thesis develops several frameworks and models that define the relationships between the various components of the adaptive bitrate selection system. This includes deriving the valid system state space, which defines the state that an algorithm can be in at any given time, determining the allowable interactions between the various components, and identifying the video quality evolution rules that optimise QoE.

Using this information, some state-of-the-art algorithms are improved and novel ones developed to demonstrate the effectiveness of the proposed approach. The result of extensive evaluations conducted both within a real-world Internet environment and with network trace shows the proposed schemes help in reducing the convergence time, startup delay, and rebuffering events, while at the same time increasing both the average and the stability of the video quality. All this is obtained without any adverse impact on the fairness among the competing players.

Acknowledgments

I will like to take this opportunity to thank all the people that in any way help me during my PhD at Lancaster.

First and foremost I will like to express my most profound gratitude to my two supervisors Dr Andreas Mauthe and Dr Christopher Edwards. Throughout this journey you have always offered me the support and guidance necessary for me to succeed as both a researcher and a scientist. Next, I will like to acknowledge the generous sponsorship from the Petroleum Trust Development Fund (PTDF) Nigeria, throughout the period of the thesis.

My thanks also go to those that I have worked with during this period, including Musab Isah, Mu Mu, Ibrahim Ethem Bagci, and Yongjun Wu (Amazon). I like to also extend my thanks to my examiners Prof. Wolfgang Effelsberg and Prof. David Hutchison.

Very import to me is those that have kept me sane throughout this period, my love Hauwa and my two angels Salmah and Ahmad. The encouragement and motivation you offered me have been unfaltering. I love you.

Last, but not the least my late parents Alhaji Sani Kabara and Hajiya Salamatu, and my mother, Hajiya Aishatu. It was your encouragement that pushed me to begin this journey, and it is your support that has sustained me to the finish line. Hajiya Baba, I wish you are here to see the product of all the late night tea. I will always miss you. And to all my brothers and sisters I say thank you for the support and encouragement.

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university.

Yusuf Sani

Lancaster University
March 2017

Contributing Publication

Yusuf Sani, Andreas Mauthe, Christopher Edwards,"*Adaptive Bitrate Selection: A Survey*", IEEE Communications Surveys & Tutorials (2017).

Yusuf Sani, Andreas Mauthe, Christopher Edwards,"*On the trajectory of video quality transition in HTTP adaptive video streaming*", Multimedia Systems (2017): 1-14.

Yusuf Sani, Musab Muhammad Isah, Andreas Mauthe, Christopher Edwards "*Experimental Evaluation of the Impact of Mobility Management Protocols on HTTP Adaptive Streaming,*", IET Networks Journal (2017).

Yusuf sani, Mu Mu, Andreas Mauthe, Christopher Edwards, "*A Bio-inspired HTTP-Based Adaptive Streaming Player*", 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Seattle, WA, 2016, 1-4.

Yusuf Sani, Andreas Mauthe, Christopher Edwards,"*Modelling Video Rate Evolution in Adaptive Bitrate Selection*", IEEE International Symposium on Multimedia (ISM), Miami, FL, 2015, pp. 89-94

Yusuf Sani, Andreas Mauthe, Christopher Edwards,"*On the Dynamics of Adaptive Bitrate Selection*", IEEE Transactions on Multimedia (under review)

Other Publication

M. Mu, W. Knowles, **Yusuf Sani**, A. Mauthe and N. Race, "*Improving Interactive TV Experience Using Second Screen Mobile Applications,*", 2015 IEEE International Symposium on Multimedia (ISM), Miami, FL, 2015, pp. 373-376

Musab Muhammad Isah, Steven Simpson, **Yusuf Sani**, Christopher Edwards, "*Towards Zero Packet Loss with LISP Mobile Node,*", International Conference on Computing, Networking and Communications (ICNC)

Awards

Winner: 2016 IEEE ICME 2016 Grand Challenge: Dynamic Adaptive Streaming over HTTP

Contents

List of Figures

List of Tables

Acronyms

ABR	Adaptive Bitrate Selection.
AI	Annoyance Index.
CDN	Content Delivery Network.
DASH	Dynamic Adaptive Streaming over HTTP.
DI	Delight Index.
FTP	File Transfer Protocol.
GOP	Group of Pictures.
GPS	Global Positioning System.
HAS	HTTP Adaptive Streaming.
HTTP	Hypertext Transfer Protocol.
ml	Machine Learning.
MPD	Media Presentation Protocol.
MSS	Microsoft Smooth Streaming.
NAT	Network Address Translation.
OTT	Over-The-Top.
QoE	Quality of Experience.
QoS	Quality of Service.

RTCP	Real-Time Control Protocol.
RTP	Real-Time Transfer Protocol.
RTSP	Real-Time Streaming Protocol.
RTT	Rounf Trip Time.
SDN	Software Defined Network.
SSIM	Structural similarity Index.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
URI	Uniform Resource Identifier.
VoD	Video on Demand.

Chapter 1

Introduction

Video streaming over data networks has been a research topic since the 1980s. In the early 1990s, video content started to be transmitted over the Internet [3]. Since then, both, the quality of the content and the variety of the video services have continued to grow. Nowadays, video traffic is the most popular service on the Internet [4]. Cisco predicts that by 2019 global video consumption will account for 80%-90% of the entire data traffic traversing the Internet [5].

A typical Video streaming service must accommodate a heterogeneous set of requirements due to the variety of content and content sources; user contexts and interests; devices and network limitations. Nonetheless, users expect the best viewing experience possible. *HTTP Adaptive Streaming* (HAS) is the most successful technology, so far, that allows content providers to cater for the requirements of this multitude of devices and the different contexts. It does this by adapting the video rate requested by a client to its context. The process through which a HAS client chooses a video rate to stream, in a given context, is called *Adaptive Bitrate Selection* (ABR).

An ABR scheme is composed of many components interacting in a non-trivial manner. To build a HAS service that maximises Quality of Experience (QoE), an ABR algorithm designer requires both the knowledge of the relationship between these components and the impact of their interactions on the resultant user experience. To contribute in this regard, this thesis presents a set of descriptive models that capture the relationships and the interactions between the different components of an ABR. The aim is to provide a video adaptation algorithm designer with the information

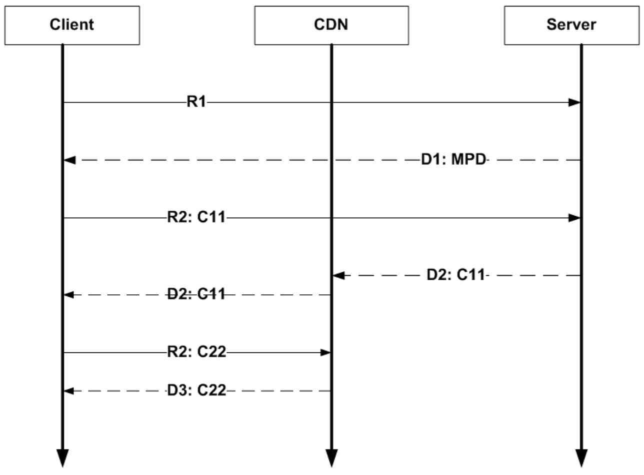


Figure 1.1: HAS Event sequence diagram.

needed to build a service that not only optimises the QoE but meets the designer's specific requirements and constraints.

1.1 HTTP Adaptive Streaming

In its standard form, HAS divides a video file into fragments, conventionally in an equal size of playtime. A fragment is called a *chunk* or *segment* (this thesis uses both interchangeably). Each chunk is encoded in multiple video rates to satisfy the requirements of the various devices and network conditions [1]. Chunks are stored together with a file called *Media Presentation Protocol* (MPD) [7] on one or more servers.

MPD is an XML metadata file containing a description of the available chunks. It provides information such as the number of different video rates per chunk, and the duration of each chunk in seconds. Figure 1.1 presents the event diagram of a typical HAS service in operation. As can be seen, when a client, first, requests a video file (for example, using an HTTP GET request) the server responds by sending an MPD. The client then uses the information contained in it to construct the Uniform Resource Identifier (URI) for the subsequent requests. Note, the URI may be directly provided or has to be constructed from a template. The parsing and processing of

[1]The criteria for providing an optimal set of video quality representations to be presented to the various streaming clients are discussed in detail in [6].

the MPD are handled by the media presentation module[2]. After the MPD has been received and the subsequent construction of the URI, a client, continuously sends a request for the next available chunk until the end of the video streaming session. The schedule and the video rate of each of the requested segment are decided based on the client's estimation of the available system and network resources. For instance, Figure 1.1 shows how for each request a client specifies both the sequence number and the video rate of the chunk being requested. In this example, R2 requests C22, which stands for the chunk number two with the second highest video rate.

Classical ABR systems select a chunk with the highest video rate lower than the measured throughput [11, 12]. When the throughput changes, the buffer level is used as an indicator of the need for increasing, decreasing or staying with the current video rate [13]. HAS services that solely rely on throughput estimation for adaptation decision are called *throughput-based*, while those that principally rely on buffer occupancy are called *buffer-based* [14]. Of course, a rate selection decision can also be made based on other parameters, such as battery power level or cost. However, this thesis is mainly concerned with only throughput-based and buffer-based players.

The primary object of a typical ABR algorithm is to ensure a high level of QoE. However, a set of metrics that fully defined what constitutes QoE in HAS services is still a topic of ongoing research. This notwithstanding, it is known that rebufferings, long convergence time, high start-up delay, frequent video quality changes, and low-average video quality are inimical to QoE [15].

Additionally, one important aspect of HAS, as can be seen from the Figure 1.1, is its ability to seamlessly work with Content Delivery Network (CDN) or any proxy server technology used by standard HTTP services. This means a content provider can cache its video on a third party CDN to save costs and reduce download latency [16, 17]. This is normally transparent to clients. A client only makes a request, whereas it is the task of server-side mechanisms to redirect the request to the appropriate proxy. Another aspect to note is that HAS is video encoding type agnostic.

[2]For detail discussion about the MPD see the following papers [8, 9, 10, 7].

1.2 Challenges

Initially, ABR schemes strictly relied on throughput estimation and selected the highest video rate lower than the measured throughput. If a download rate is more than the playback rate, a high video quality without a rebuffering event can be achieved [18, 19]. It gradually became evident that throughput estimation alone would not be a sufficient factor in making effective video adaptation decision. This is because the available capacity is time-varying, and an accurate bandwidth estimation above the HTTP layer is hard to achieve [20]. Consequently, those video rate selection algorithms that solely depend on throughput estimation for adaptation decisions are found to result in 'unnecessary' rebufferings [14], undesirable variability of video rates [20], and a sub-optimal video quality [20].

Various attempts have been made to improve the video quality adaptation decision. First, a weighted average is used to smooth out the estimated network capacity [13]. However, this scheme is known to reduce the responsiveness of an algorithm [13]. Further, supplementing throughput measurements with information about the playback buffer [21, 22], or even replacing it altogether has been proposed as a remedy [14, 23]. Figure 1.2 presents a simplified activity diagram; we derived from a study of some state-of-the-art ABR algorithms, such as those found in [14, 19, 24]. As can be seen, the flow is one way and bottom-up. Typically, a service starts by measuring the throughput and then assumes that throughput dynamics is the sole determining factor of buffer state changes. The adaptation module takes either one or both the buffer level and the estimated throughput as input. The adaptation module will then, without feedback from a user experience module, decides which video rate optimises the user experience.

However, in reality, components of an ABR scheme exhibit non-trivial interactions. For example, it has been shown in [20, 25] that buffer dynamics have a direct impact on the throughput perceived by a client, which makes the relationship both cyclic and anisotropic. This is because an ABR requests chunks discretely. Also, when the buffer reaches a particular threshold, download ceases until a certain amount of

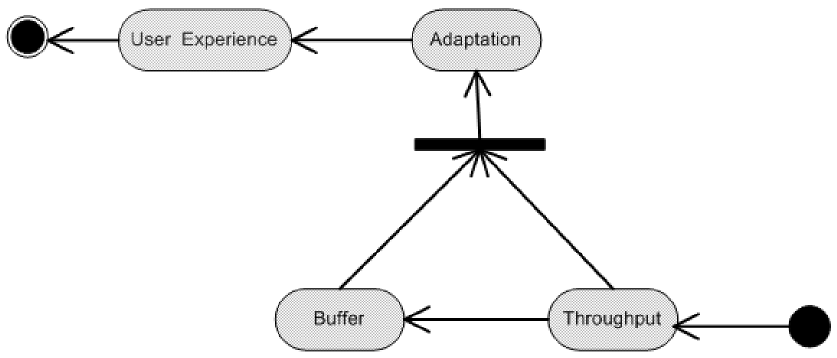


Figure 1.2: The activity diagram of a typical HAS service

content has been consumed to free up buffer space. This results in an *ON-OFF* traffic pattern [26, 27]. Whenever the OFF period is activated the TCP throughput goes to zero, and it may take some time to recover when the ON period starts. Another example of this non-trivial relationship is between the video bitrate and perceived video quality. It is a well-known fact that the relationship between these metrics is asymmetrical [28, 1, 29]. Furthermore, other factors e.g. stability of requested rate, have a direct impact on the perceived quality. Therefore, increasing the video rate without taking into consideration the prevailing QoE metrics may not always enhance the user experience [28, 1, 29].

Thus, optimising QoE, which is the central theme of any ABR scheme, can only be accomplished by not only *carefully crafting* individual components of ABR but by also taking into consideration *the relationship and the interaction* amongst the various components.

1.3 Problem Statement

Considering the current the challenges facing the state-of-the-art ABRs, there are several research questions related to this topic. The aspect addressed by this thesis is summarised as follows: *The quality of adaptive video streaming services can be improved by incorporating, at the algorithm design phase, a systematic model of the relationship between the various ABR components.* In other words, the objective of this research is to model the relationships between the different components of the adaptive bitrate selection system so that a set of states, interactions, and video quality evolution rules that optimise QoE can be identified. Armed with this information, an algorithm designer can better build services that maximise user experience. This objective can be achieved through the realisation of the following set of goals:

1. *To identify the valid system states, and the pattern of transition between the identified states that maximises user experience.* Variation in the streaming context, usually, forces the adaptation logic to change the bitrate of the requested chunk. This results in a continuous change in system state. First, what combination of factors constitute a valid system will be identified. After that, a descriptive model of the video rate map that combines all stages of the video rate evolution will be developed.

2. *To develop a model of the behaviour and the interactions between the various functional components of the adaptive bitrate selection module.* The dynamic nature of an ABR system state requires that a change in video rate must guarantee that this does not result in a drop of user experience. To achieve this, first, the valid components that constitute an ABR will be identified. Then, a framework of the relationship between them will be elucidated. From these, a model that captures the interactions between the various functional components of ABR will be developed.

3. *To develop appropriate algorithms to demonstrate the effectiveness of the model.* Since the proposed model is a descriptive representation of the relationship between the various components of an ABR, it has to be demonstrated how it can be used in practical systems. This will be done through either modifying existing algorithms or through developing new ones.

1.4 Research Scope

The subject matter of this thesis is client-side HTTP-based adaptive video streaming. The theme is restricted to cover the VoD as opposed to live streaming. Furthermore, we assume video delivery is over the best effort Internet and not over a managed network. Building an efficient video rate adaptation logic is a non-trivial issue since there is a variety of requirements from various stakeholders, often at odds with each other. This thesis argues that building an effective adaptive streaming service requires an understanding of the relationship and interactions between the various components of the system. Since the aim is to provide an ABR algorithm designer with a tool to help him/her in his/her craft, only descriptive modelling techniques are used. Any predictive scheme used will come from the algorithms using the models. While this restricts the scope of the design, this is outweighed by the advantage of providing a broad spectrum of applicability.

Furthermore, in deriving the models, no user study has been conducted but rather existing datasets and validated findings have been used since all that is required for building the models can be derived from the existing complementary research. Finally, it should be noted that any service built in the course of this research is a proof-of-concept implementation. This helps reduce the development cycle and allows for statistical analysis tools to be easily embedded into the implementation.

1.5 Thesis Organisation

The thesis is organised into six chapters. Following this chapter is Chapter 2, which explores the background and the related work. It starts with a historical account of Internet-based video streaming service, then followed by an overview of the HTTP-based streaming services. The chapter concludes with a detailed discussion on the impact of context management and the QoE on the performance of the adaptive streaming service.

Chapter 3 starts by motivating why a careful selection of system state space is important. Then defines the valid states. A model of the video rate evolution trajectory is then developed. After this, relevant state-of-the-art ABR algorithms are modified to work with the new model. The chapter concludes with a discussion on the performance evaluation results.

This is followed by the Chapter 4, which improves the framework presented in the previous chapter. Using system dynamics methodology a generalised model of the dynamics of adaptive video streaming is presented. From this, a new framework that equips the adaptation module with both the system level information and QoE metrics is developed. After that, the proposed model of the video quality dynamics in both a resource abundant and constrained context is discussed.

In Chapter 5, the thesis presents a cooperative algorithm built on top the framework earlier discussed in the previous chapter. A prototype is designed and implemented, followed by a detailed evaluation of the proposed service. Chapter ch:dis discusses the different aspects of the research and reflects on the technical contributions of the previous chapters. Finally, the thesis concludes in Chapter 7 with a summary, contributions, and a discussion of future work.

Chapter 2

Background and Related Work

The delivery of video traffic over the Internet has remained an active research topic despite the substantial effort that has been invested in it during the past three decades [30, 3]. This is primarily because the Internet has been built to provide best effort services, and was essentially not designed to ensure Quality of Service (QoS). However, for video streaming services to operate satisfactorily, a certain level of QoS has to be guaranteed e.g. packet loss and delay must not exceed a certain limit.

Several attempts have been made to extend the Internet with QoS capabilities [31, 32] that assure the quality of the video delivery service mechanisms through specialised network architectures and protocols [33, 34]. However, these efforts have not resulted in a wider deployment within the standard networks, which is, possibly, because of their complexity; the increase in capacity and the penetration of broadband; improvement in the efficiency of video compression techniques; and the prevalence of adaptive video access and delivery mechanisms.

Currently, the approaches proposed for video content distribution over the Internet can be categorised according to the type of network management used for the set-up and transmission of the video. Video can be implemented either over managed or unmanaged networks [35]. Managed services are typically used by cable-TV and IPTV services. These services are provided over a dedicated network infrastructure[1] that ensure QoS through techniques such as DiffServ [31, 36]. Due to this, the managed video services usually ensure a high video quality [37]. Nevertheless, managed networks are expensive to setup and maintain. Hence, those services that depend on it are typically provided by big organisations such as ISPs, Telecoms or cable companies. Video services that are delivered over unmanaged networks are called Over-The-Top (OTT) services. The unmanaged services are mostly offered to the end users via the best-effort Internet either by the content providers directly or through third parties e.g. CDN service providers. Since OTT services require no specialised or dedicated infrastructure their setup and maintenance costs are relatively low.

In this chapter, we provide a review of the some of the most important research activities regarding client-side Video on Demand (VoD) OTT services. The chapter starts by presenting a short historical evolution of the Internet video. It then discusses the impact of transport layer protocols for video delivery. From this, all attention is diverted to the theme of this thesis. A detail of HTTP adaptive streaming is then presented, followed by a discussion of the ABR module, which is divided into three subcomponents, namely: resource estimation, chunk request scheduling, and adaptation. A review of each subcomponent and how it interacts with other components and its operating environment is presented.

2.1 Evolution of Internet Video

The architecture of OTT video streaming has passed through some phases. Each phase relied on particular assumptions of how transport layer protocols of the network stack affect the quality of media content and the playback rate. In the beginning, a video file was downloaded like any other large file from a remote server using File Transfer Protocol (FTP) [30]. This can take between a few minutes to many hours depending on the file size and network condition. The design of this architecture is simple since it only involves a commodity web server, and a client needs only a player that can decode the downloaded file. The method is robust because it uses a reliable

[1]Note, these can be virtual networks as well as physical networks.

transport protocol, Transmission Control Protocol (TCP). Consequently, the content quality is high. However, since video files tend to be large even if efficiently encoded, a complete download involves significant disk storage requirement at the client side. Furthermore, this method naturally precludes any element of interactivity while a file is in transit.

The second phase is called *progressive download*. It also considers the stored media as a single file to be downloaded at once, but with one significant difference: it is a video streaming scheme. This means when streaming using progressive download scheme playback starts before the whole video file is downloaded. A client is expected to buffer the incoming packets as they arrive. As soon as a predefined amount of content is buffered, playback starts [38]. This significantly reduces the initial start-up latency to a few minutes, in the worst case. However, it wastes bandwidth when the user decides to terminate the streaming session before it ends. Another disadvantage of progressive download is that it does not support live streaming [7]. Furthermore, when a network capacity drops, a viewer either stops the current session and starts a new one, from the beginning, using lower video rate, or persists and suffers rebuffering events.

Video has an intrinsic transmission rate, which makes video streaming delay and bandwidth fluctuation sensitive. However, the Internet does not provide sufficient QoS guarantee. To ameliorate the impact of this lack of QoS guarantee, video streaming services began to adapt the quality of video to the network conditions [39, 40, 41] so that the bitrate of a delivered video stream matches the available bandwidth. The first generations of adaptive video quality services re-encode the stored video on-the-fly [42]. However, this technique is, generally, not scalable, because encoding video content is a CPU intensive exercise [40]. Another approach is based on the layered encoding [43, 44, 45]. Layered coding allows the encoding of a video into several layers, made up of a base layer (representing the least quality level), and some enhancement layers, with each enhancement layer improving the viewing quality [43]. In this scheme, the higher the available bandwidth, the more layers that are delivered to the end user. In the third option and the most recent, a server provides video in multiple adaptation sets. Each representation in the adaptation set is encoded into multiple versions using a combination of encoding configurations, such as resolution and bitrate, which are stored as a series of addressable chunks (segments). Using a manifest file, a player continuously requests a chunk of appropriate bitrate based on the level of the available resources [13, 20, 7]. Regardless of the technique used in the delivery of video content, the transport layer protocol used for the transmission can only be either the User Datagram Protocol (UDP) or TCP.

2.1.1 UDP-Based Video Services

UDP is a connectionless protocol that provides unreliable delivery service. It has no flow control mechanism. Hence it is suitable for applications that put more emphasis on promptness over reliability. However, UDP is prone to packet loss. To effectively stream video, UDP-based services require additional protocols to supplement UDP. First is Real-time Transport Protocol (RTP) [33], which is responsible for providing time-stamps and sequence numbering to the transported stream. The Real-Time Control Protocol (RTCP) [33] then is used to gather statistics about the state of media transfer during a particular streaming session. A client will normally communicate these statistics to a server that uses it to adapt its response.

At the application layer, UDP-based video streaming services mostly rely on Real Time Streaming Protocol (RTSP) [34]. The RSTP is designed to be stateful. A server maintains session states, which enables it to track the client's states throughout a period of a connection and unlike Hypertext Transfer Protocol (HTTP), RTSP allows two-way communication between client and a server, which enables the server to provide value-added services such as rewind and forward. It should be noted that nothing technically stops an HTTP-based service from being built on top of UDP. In fact, some researchers have done so [46]. Nonetheless, these additional protocols require specialised a media server. Therefore, deployment and maintenance of UDP-based video streaming service are relatively complex and expensive. Additionally, UDP based services are not Network Address Translation (NAT) friendly, and are often blocked by the default firewalls settings [18, 47].

2.1.2 TCP-Based Video Services

TCP is designed to provide a connection-oriented, reliable, byte-oriented end-to-end communication service [48], with a flow control mechanism that ensures a sender does not overwhelm a slower receiver [49]. Importantly, it has an inbuilt congestion control mechanism. For this, TCP tries to adjust its sending rate after a serious packet loss occurs. TCP has two modes to deal with packet losses. The first mode is activated when it receives a request for retransmission of a number of lost packets. Here, the TCP enters what is called the fast recovery phase. After successful retransmission of the lost packets, the window size is reduced to half of its previous value and, after every Rounf Trip Time (RTT), it is increased by one unit, effectively reducing the sending rate to half. The second mode is activated as the result of a retransmission timeout. More drastic action is taken because TCP assumes that there is severe congestion in the network. The window size is reduced to one unit, and then, after every RTT, the window size is increased exponentially until it reaches the slow-start threshold value. Subsequently, the window size is increased linearly. Whenever the slow-start mode is activated, the throughput is reduced virtually to zero.

Obviously, these protocol features increase packet delay, jitter, and throughput fluctuation. As a result of this, until recently, TCP was thought to be unsuitable for an efficient video streaming service. Currently, most of the assumptions made regarding the unsuitability of TCP, as a transport protocol for video streaming, have been put into question. First, it is now known that bandwidth fluctuations can be smoothed-out by pre-buffering content at the receiver side [50, 51]. Secondly, Wang et al. [52] have shown that if the TCP achievable throughput is twice the bitrate of the targeted video plus few seconds of start-up delay, TCP can ensure a good streaming performance.

Furthermore, the use of UDP as the transport layer protocol implicitly assumes that video is somewhat packet loss tolerant. Certainly, this is only correct to an extent. Most modern video compression techniques, such as H.264 [43], rely on compensation prediction algorithms. This results in interdependence between successive frames. Imagine a lost of I-frame; it will be difficult for a player to reconstruct the affected Group of Pictures (GOP) successfully. Consequently, retransmission of lost packets will be desirable [53]. This is why elaborate error recovery and concealment schemes always accompany UDP-based services [54]. In fact, even the RTP provides some level of retransmission [39, 40] however, TCP requires no such additional features.

Consequently, the choice of transport layer protocol becomes a trade-off between visual degradation and video stall [55]. This fact notwithstanding, it has been shown in [55, 56] that 'given any bottleneck bandwidth' streaming with TCP outperforms UDP in terms of visual fidelity. Thus, TCP-based streaming services are preferred by users.

2.1.3 HTTP-based Streaming Service

Given the fact that HTTP is the most widely used protocol for content delivery over the Internet, it is natural for it to be used when streaming video. The progressive download paradigm is entirely built on top of it. For example, in the progressive download, a player will usually send a one time HTTP GET request to a server and wait for a response, on the receipt of the response the streaming session begins. Streaming on top of HTTP has many advantages. The first is that HTTP is well understood and easy to set-up. Furthermore, Internet infrastructure over time has been tuned to efficiently handle HTTP traffic, for example by using HTTP with a streaming technology the existing content delivery technologies designed for ordinary web usage can be reused [57]. Also, HTTP is usually not blocked by firewalls.

HTTP Adaptive Streaming (HAS) is the most recent video quality adaptation scheme that employ HTTP, and uses TCP as its transport technology [9, 7]. Presently, almost all the standardisation bodies that have an interest in media delivery have either separately or jointly standardised it. For example, IEFT [9], 3GPP [10], and the Open IPTV Forum (OIPF) [58]. These efforts led to the most widely accepted standard MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [59]. There are also some commercial implementations, e.g. Microsoft Smooth Streaming

(MSS) [60], Apple's HTTP live Streaming (HLS) [9], and Adobe OSMF [61]. Also, most of the well-known video streaming services like Netflix [62, 20] and YouTube[62, 63] are known to be using HAS based streaming services.

HAS allows a content provider to store video in several resolutions, with each targeted at a customer segment. Additionally, a resolution is encoded in multiple bitrates. Each bitrate is carefully chosen to fit a particular streaming context. To allow a player to adapt its media stream to a changing context dynamically, each video file is segmented into, usually, equal temporal size chunks. A client is provided with a manifest file, such as the Media Presentation Protocol (MPD) standardised in MPEG-DASH, which contains all the necessary information needed to request a chunk from a server. A client continuously monitors and estimates its capabilities, and then requests a chunk with a video rate it deems appropriate. While the media presentation has been standardised, the adaptation function is determined by individual ABR designers.

2.2 Adaptive Bitrate Selection

To adapt video quality to a particular context, system performance parameters such as available bandwidth, buffer size, and battery life, are measured. The choice of which parameter becomes a situational indicator depends on the QoE metric that an ABR intends to optimise. ABR then uses the measurement result in making a decision on the schedule and the profile of a chunk to be downloaded.

A typical ABR operates in two states [26, 64, 62]: *Buffering* and *Steady* state. At the buffering state, conventionally, a player starts requesting the lowest video rate. After which the ABR may try to fill its buffer as quickly as possible [14, 23]; raise the video rate as fast as possible [26, 64]; or combine the two approaches [65]. If the goal is to fill the buffer, the chunk request rate has to be maximised by aggressively downloading low bitrate chunks. This technique is meant to reduce the start-up delay and protect a player from buffer under-run [62]. But this may result in low video quality at the start. However, if the goal is to quickly ramp-up the video rate, chunks with higher rates are requested. This technique increases the start-up delay. The buffering phase is activated when there is an imminent possibility of buffer depletion. This is typically the case at the start of a streaming session or after a rebuffering event. It is important to note that a player normally does not wait until the end of the buffering phase before a playback begins. All that is required is a sufficient amount of data in the buffer to absorb fluctuation in the network capacity. This can be achieved when the buffer size reaches a predefined target [66]. For how the threshold value is decided on see Section 2.3.2.

When the buffer size reaches the threshold value, steady state mode is activated. In this phase, the goal is to maximise the video rate of the requested video segments, such that only chunks having the highest video rate the available system capacity can sustain are targeted. One important feature of the steady state phase is called *periodic download* [26, 66]. Periodic download is a technique that allows a client to download a chunk and then pause for some time before downloading the next chunk. At the ON period, the speed of a download is only constrained by the TCP throughput, while at the OFF period no data is downloaded [26, 19]. The periodic download guarantees an inflow of content into the playback a buffer without causing an overflow.

To better manage the complexity of ABR, we decompose it into three subcomponents, as can be seen in Figure 2.1. Each unit is associated with a particular function that an ABR is expected to render. It is worth noting that subcomponents need not be necessarily co-located in one machine. In fact, any of the modules can be located in separate systems. Even though not all application designers will decouple ABR this way, it is still expected that even a purely monolithic design contains all of the three main components as follows:

- resources estimation module,
- chunk request scheduling module,
- adaptation function.

As can be seen in Figure 2.1, the chunk scheduling function takes as inputs the time the last chunk download finished and the current buffer level and then decides when a

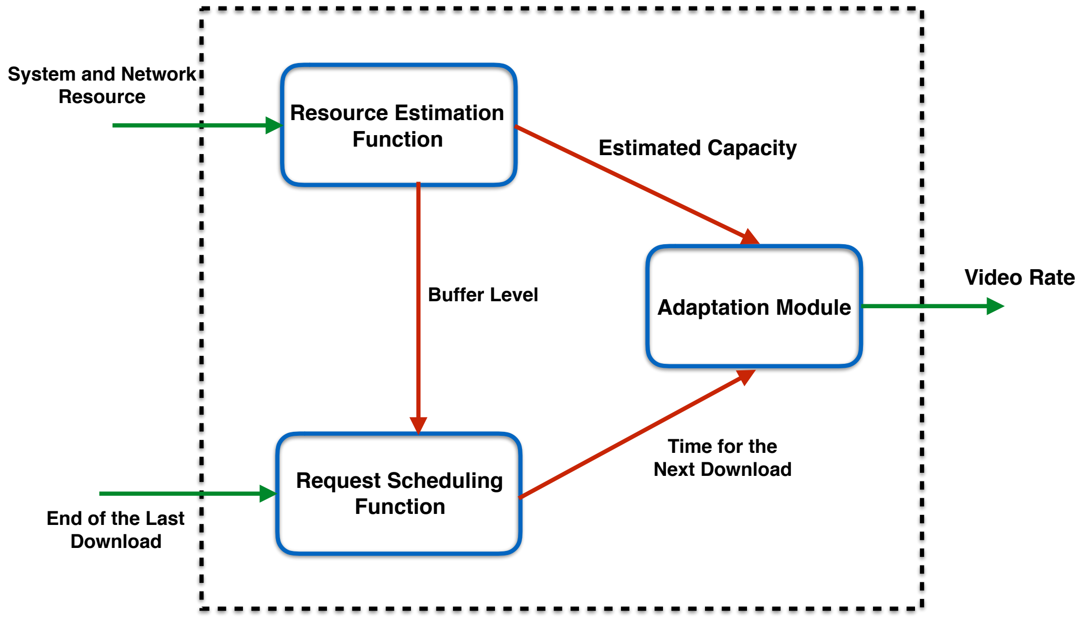

System and Network Resource
Resource Estimation Function
Estimated Capacity
Buffer Level
Adaptation Module
Video Rate
Time for the Next Download
Request Scheduling Function
End of the Last Download

Figure 2.1: ABR framework.

chunk is going to be requested next. The adaptation module determines which profile of the chunk should be downloaded, based on feedback from the resource estimation module. The framework does not consider the encoding and uploading of content to a server since rate adaptation has no influence on their timing with regards to the VoD services.

2.3 Resource Monitoring and Measurement

An ABR is expected to monitor and measure the resource of interest. This task can be implemented at the server-side, the clientside, or somewhere inbetween. The appropriate location depends on which performance parameter an ABR relies on. The client-side ABRs distribute the task of resource observation, hence are more scalable. Since for content providers, targeting a large number of clients is their primary goal, and some of the most critical parameters may be better observed closer to the client e.g. last-mile bandwidth, buffer occupancy, and power level, monitoring and resource

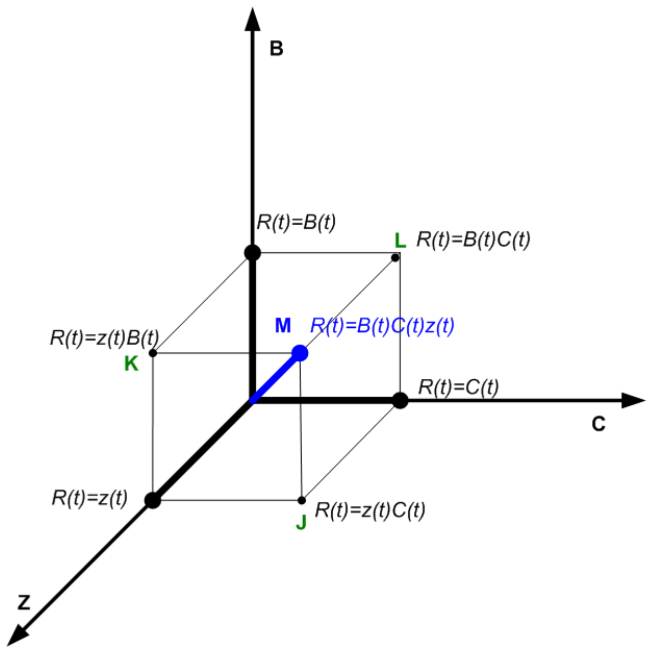


Figure 2.2: 3-dimensional representation of all the possible forms of $R(t)$.

estimation is usually implemented at the clientside.

However, solely relying on client-side observations results in an opportunistic ABR [25]. To address this challenge, a central control plane has been proposed in [67] that aggregates measurements from many clients. This ensures that ABR is globally optimising performance across all clients. In [1] a control plane is used to orchestrate the monitoring and measurement process of video streams in a network. The goal here is to ensure network-wide QoE fairness. In [68, 69] a network control plane, with the aim of maximising network-wide QoE and bandwidth utilisation, is proposed.

For the purpose of resource utilisation, ABR can be represented as a function $R(t)$. The function $R(t)$ can be classified according to the input parameters. Some of the parameters often used to characterise $R(t)$ are throughput [19], buffer occupancy [14], power level [70] and cost [71]. Typically, $R(t)$ takes n parameters as input where $n \geq 1$. To represent all the possible classes of $R(t)$ we need an n-dimensional plane, but this will be difficult to plot, therefore, we use a 3-dimensional plane as can be seen in Figure 2.2. The principal parameter an ABR relies on is called the ***main factor*** with weight γ, other factors are then referred to as ***adjustment factors***, with combined weight of $(1 - \gamma)$. When no parameter dominates, all should be considered as adjustment factors.

Let us assume that $C(t)$ denotes throughput, $B(t)$ represents buffer occupancy, and $Z(t)$ is any other parameter that may be used in modelling $R(t)$ (e.g. cost or battery life). Hence, in the Figure 2.2 $C(t)$, $B(t)$ and $Z(t)$ axes represent $R(t)$s relying on one parameter, as the case may be. On any plane between any two axes there are various possibilities of *mixed-mode* ABRs. For example, point K, where $R(t) = \gamma B(t).(1-\gamma)Z(t)$, represents an ABR that relies on both the buffer occupancy and another factor (e.g. battery level). Point L, where $R(t) = \gamma B(t).(1-\gamma)C(t)$, is an ABR that relies on both throughput estimation and buffer occupancy. Any one of the two metrics can be the main factor or the adjustment factor. Point M with $R(t) = \gamma B(t).(1-\gamma-\beta)C(t).(1-\gamma-\theta)Z(t)$ where $\gamma+\beta+\theta = 1$ represents a typical $R(t)$ that relies on more than two metrics, for instance as discussed in the work of Tain et al. [72].

ABRs that have throughput estimation as their main factor are called *Throughput-based* ABRs, while those using buffer occupancy as their main factor are called *Buffer-based* ABRs. *Power-based* ABR use battery level as their main factor. It should be noted that an ABR that relies on only one factor is a special case of a mixed-mode ABR with the potential adjustment factors having zero weight.

2.3.1 Throughput-Based ABR

Throughput-based ABRs try to estimate the available network capacity, which is the average unutilised capacity over a specific time interval [73]. Regardless of the underlying technology or the transport protocol used for any content transmission, the available bandwidth is time-varying [74]. When TCP is employed as the transport layer protocol (as in the case of HAS) the variability is exacerbated by the protocol specific characteristics (e.g. TCP slow start and congestion control) see Section 2.1.2 for more detail.

Usually, throughput-based ABRs equate the available bandwidth with the measured TCP throughput. Furthermore, the monitoring and estimation of the available network capacity are often done above the HTTP layer, which results in not very accurate information [20]. Moreover, because of the discrete nature of HAS based systems, a resource estimation function can only estimate per-chunk throughput. This can easily be obtained by dividing the amount of data (the size of a chunk in bytes) downloaded by the duration of the download. The idea is to use the throughput of a recently downloaded chunk as a rough estimate of the current network conditions. But throughput derived from a single chunk results in a significant variability in the video quality. Due to this, and the difficulty in accurately estimating throughput above the HTTP layer, various techniques described as follows are used to improve the quality of the measurement.

2.3.1.1 Estimation Techniques

Long before HAS was proposed, Prasad et al. argued that any meaningful available bandwidth estimation technique requires a time averaging of the instantaneous estimates over a time interval [74]. A variety of averaging methods has been used to estimate the available bandwidth to overcome the issues related to per-chunk estimation, and the instability of video rate this creates. In [19], the harmonic mean is used to smooth-out the estimated instantaneous throughput. One reason for this choice is the robustness of the harmonic mean to large outliers. Qiu et al. [75] employed an exponentially weighted moving average. By doing so, they are not only able to incorporate historical estimates into the current estimate but also exponentially reduce the significance of the historical data as time passes.

However, it is a well-known fact that smoothing techniques tend to inhibit the responsiveness of an algorithm, which may cause a late reaction to a significant variability, perhaps, that requires an urgent action, for example, a late response to a large throughput decrease, which can result in a buffer underrun.

2.3.1.2 Reliability of the Estimate

Still, an open issue in ABR research is the extent to which the throughput estimates reflect the actual state of the available bandwidth.

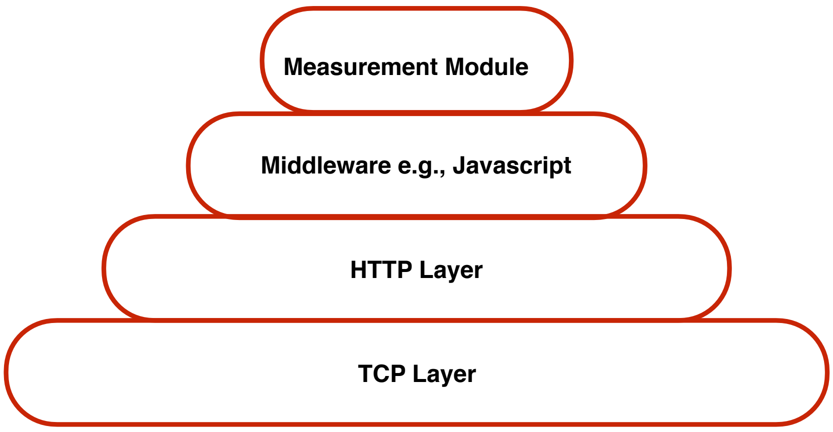


Figure 2.3: HAS protocol stack

Any throughput measurement that is done at the application layer, at best only represents the throughput of the underlying TCP. It is argued, in [73], that equating the available bandwidth with a bulk TCP throughput is a fallacy. Since TCP throughput depends on many factors (including socket buffer sizes at the sender and receiver, nature of the competing cross-traffic, round-trip time, loss rate, the nature of TCP congestion control).

Similarly, Li et al. in [76] have argued against equating the TCP throughput observed at the application layer with the available bandwidth. They showed when clients compete for the available resource, the presence of competing applications and the discrete nature of the HAS downloads make it difficult for a client to perceive its fair share of the available bandwidth correctly. As a consequence, the client underestimates the throughput, which results in an under-subscription of the available bandwidth and video rate oscillation. The latter is known to impact user experience negatively [13]. As a remedy, they propose PANDA: a probe and adapt technique. The algorithm somehow mimics the congestion control of TCP but at the application layer. It uses TCP throughput as an input when it is an accurate indicator of the fair-share of bandwidth. They argued that this happens when a network is congested, and the off-interval is absent. Otherwise, the scheme probes the network by incrementing the sending rate, and backing-off when congestion is detected.

Figure 2.3 presents the classic protocol stack of HAS. As can be seen, because the measurement module sits on top of a middleware e.g. Javascript, which in turn sits on top of the HTTP layer, its result will hardly be the actual TCP throughput. This is confirmed by [20] Huang et al. They investigated three video streaming services, i.e. Hulu, Netflix and Vudu, and found that an accurate client-side bandwidth estimation above the HTTP layer is hard. They argue that any rate selection based on such an inaccurate estimate will trigger a feedback loop that leads to an undesirable variability and unnecessary reduction in video quality. Furthermore, they observed that the cause is a lack of information since the HTTP layer does not get a continuous high-fidelity feedback about the fair share at the bottleneck. The paper stresses that determining the fair share of the bandwidth available at a bottleneck link is precisely the role of TCP. To deal with these issues, the paper made the following suggestions: first, an ABR algorithm should improve the information flow from TCP to the HTTP layer, which will ensure that TCP has a chance to reach its steady-state (e.g. by increasing the segment size). Second, ABR algorithm can just rely on buffer state changes for chunk selection (see Section 2.3.2 for more detail).

To improve the flow of information along the video streaming service stack shown Figure 2.3, some attempts were made on cross-layer throughput estimation. Guibin and Yong [21] use machine learning techniques developed in [77] to predict the achievable throughput. The method uses the support vector regress algorithm [78] to train the throughput prediction model with network layer information like packet loss, delay, and RTT. In [79, 80] Ramamurthi et al. use physical layer 'goodput' to complement the application layer estimate. This resulted in an improvement in the perceived video quality and a reduction in the rebuffering frequency.

2.3.1.3 Impact of TCP Dynamics

The extent to which a throughput measurement module fails to provide an accurate estimate is not the only reason for the failure of clients to estimate their fair share of the available bandwidth. TCP congestion control dynamics play a significant role, also. There are two main options to solve this problem, either by modifying the congestion control mechanism of TCP, as recommended by Kupka et al. [27]; or reducing the causes of the unnecessary activation of the congestion control mechanism. This can be achieved by making a streaming service more resilient to packet loss. Generally, the HAS community pays most of it attention to the latter option.

Nguyen et al. [81] proposes multiTCP, an application layer algorithm that improves resilience against short-term TCP throughput fluctuation. They demonstrated that for any packet loss event, the reduction in throughput when two TCP connections are used is four times less than if one TCP connection is used. In summary, the amount of TCP throughput reduction is inversely proportional to the square of the number of TCP connections employed.

Kuschnig and Hellwagner [82] use multiple HTTP/TCP connections to stream HAS content. The scheme is found to be resilient to packet loss, and therefore, can reduce throughput fluctuation. While in [81, 82] all streams share a bottleneck link In [83] attempts to improve the resilience of the system by concurrently fetching chunks from multiple servers is made. The scheme continuously estimates the bandwidth of each stream from all servers. A software agent decides which representation will be requested based on the smoothed version of the estimate. The agent requests a slice of the chosen chunk from each server concurrently in proportion to the estimated capacity of the corresponding server. The result of their experiment shows a reduction in the video rate variability at no extra bandwidth.

2.3.1.4 Impact of Traffic Pattern

Usually, an ABR, as shown in Section 2.2, requests chunks discretely, which leads to an *ON-OFF* traffic pattern resulting in a naturally bursty behaviour. Fine-tuning bursty traffic is a well-established technique used in many bandwidth estimation applications [84]. This technique is only required since most bandwidth estimation schemes work best under the assumption that network traffic has fluid flow characteristics [85]. It should be noted that this assumption may not be valid always. The authors of [73] have argued that disregarding the bursty nature of traffic is one of the pitfalls of bandwidth estimation. Nonetheless, since HAS traffic does not appear as a continuous flow, one needs to evenly space the traffic to give it the appearance of fluid flow. One technique used to achieve this is traffic shaping. Traffic shaping is a technique used to regulate the flow of a network traffic by delaying or prioritising a flow of certain packets, which gives the flow a near constant output rate [86]. Traffic can be shaped at the server-side, the gateway, or the client-side.

Akhshabi et al. [87] implement a server-side traffic shaping technique that is not player-specific. It is aimed at neutralising the OFF period in a steady state. The first thing the shaping module does is to detect oscillation. This happens when the profile of contiguous requests frequently alternate. The next step is to limit the throughput of a chunk to its encoding rate. In other words, the download duration of the chunk will now be equal to the chunk playout duration. Meanwhile, they have already set the inter-arrival time to the chunk duration. Limiting the chunk throughput will therefore effectively remove the OFF period. Hence, the player remains in the ON period even though it is in a steady state. The results of experiments show that the mechanism reduces the instability. In summary, the clients are better able to estimate their fair share of the available bandwidth. Though, the authors argue that this is a reactive mechanism, and have a reduced execution overhead on the server, the web server still requires modification.

Houdaille and Gouache [88] propose a traffic shaping mechanism implemented at the home gateway. The technique allows for bandwidth arbitration by first determining the bandwidth requirement of each of the streaming clients, and then constrains the clients to stay within their allocated limit. Villa and Heegaard [85] propose a client-side traffic shaping technique that does not involve the server or the gateway in any way. They interleave requests from different clients. When traffic consists of low profile chunks, the effect of shaping is realised by increasing the inter-arrival time, while when the traffic consists of high profile chunks the traffic shaping is achieved by increasing the segment fetch duration.

2.3.2 Buffer-Based ABR

Buffer serves several purposes in streaming systems. Initially, it was introduced to absorb throughput variability [89] so as to ensure that the time restrictions of contin-

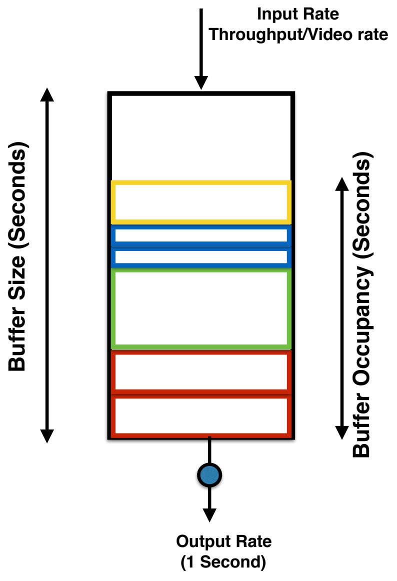


Figure 2.4: Buffer dynamics

uous media (i.e. deadlines) are met. Moreover, for a player not to run out of content the rate at which it consumes data should be at least equal to the rate at which the content arrives, which is not always the case. Hence, to avoid buffer underrun, a certain amount of data is *buffered*. This ensures that the HAS client will continue playing from the prebuffered content for at least a time equivalent to the duration of the buffered video [90].

2.3.2.1 Buffer Dynamics

In the early days of video streaming, before scalable video or HAS video streaming services were available, a video content was required to have a uniform quality. In these systems, a buffer capacity is specified in bytes. To derive the temporal size of a buffer occupancy, one just divides the size in byte by the average video playback rate [2]. However, with the introduction of video services where the video rate could be adapted to the changing conditions (e.g. scalable video coding or HAS services) this is not necessarily the case anymore. For instance, the buffer of a HAS client (as can be seen from Figure 2.4) at any given time can store multiple chunks of which each

[2]The average video playback rate is used because variable bitrate coding is the most widely used encoding scheme.

chunk can be of any video rate between the lowest and the highest available video rates. To the best of our knowledge, currently, there is no straightforward method of doing conversion of the size in bytes to time. Hence, a buffer in HAS services is generally calibrated in time [21, 19].

The rate at which content is fed into the buffer depends on both the download rate and the video rate of the chunk being downloaded (see Figure 2.4). Furthermore, the buffer is drained at the rate of one second of playback every second of real time. Hence, if the ratio of the download rate to the video rate (i.e. the play-out-rate) is greater than *one* buffer occupancy grows, and if it is less than *one* it shrinks. This effectively means that to maintain the buffer occupancy at a given level, before a chunk is played the next chunk must have arrived. Moreover, the lower the current buffer occupancy becomes, the more likely it is that the video will freeze. In fact, the authors of [91] have found that the probability of buffer starvation decreases exponentially with respect to the initial buffer level.

2.3.2.2 Buffer as a Feedback Signal

Until recently, ABR algorithms divide buffers into logical segments $S_0, S_1, ..., S_{n-1}$ with $B_1 < B_2 < ... < B_{max}$ as thresholds. The logic behind the segmentation is to allow the pertinent ABR algorithms to behave differently in each buffer segment [22, 21, 92]. The least number the buffer can be segmented into, is two. The first segment S_0 is an area from when the buffer is empty to a threshold point B_1 (this may be any point less than the maximum buffer size). The second segment S_1 is the area from the threshold to the maximum buffer level (B_{max}). A typical example of the work that uses this scheme is [21]. However, others such as MSS [64] and Miller et al. [22] divide the playback buffer into three segments, called *panic*, *low*, and *upper* level.

The use of buffer as an adjustment factor is based on the assumption that the main factor in any rate adaptation is channel capacity. However, the knowledge of network capacity, as discussed in Section 2.3.1, is at best imprecise. Upon realising that relying on TCP throughput only results in an ABR that is unstable [76], unnecessarily rebuffering [20], requesting sub-optimal video rates [13], and unfair [20] researchers started prioritising buffer occupancy when making adaptation decision. Tian and Liu's work [21] is an important milestone in shifting towards buffer-based ABR. The authors used buffer occupancy as the main factor with TCP throughput as an adjustment factor in their ABR. The algorithm they propose relies on three buffer related properties, i.e. (i) buffer size adjustment factor, (ii) buffer trend adjustment factor, and (ii) video chunk size adjustment.

To the best of our knowledge, the first ABR that solely relies on buffer state changes, for the purpose of rate selection, is the work of Huang et al. [14, 23]. The authors argued that the ultimate aim of any rate adaptation algorithms is to control a playback buffer. And its most important task is to prevent rebuffing. Furthermore, they argued that buffer dynamics contain a lot of information that is sufficient for an ABR to make decisions without recourse to any other parameter. Therefore, they conclude 'if it is the playback buffer we are controlling, then why not measure and control its occupancy directly?' [14].

The algorithm proposed in [14] works as follows: (a) provided the start-up period is passed, the current chunk bitrate is increased to the next level if the rate suggested by the rate map exceeds the next higher available video rate ; (b) the current representation is reduced to the next lower rate if the rate suggested by rate map is lower than the next available quality level lower than the current level; otherwise, (c) the current video rate is maintained. Further, they suggest that a segment of the buffer is reserved for the start-up period, called a 'reservoir'. While filling this reservoir, only chunks with the lowest video rate are requested. They analytically show that the algorithm will never rebuffer provided the network capacity is more than the minimum video bitrate, and will always converge at the video rate that matches the network capacity.

2.4 Chunk Scheduling

Chunk scheduling is the process of deciding when to dispatch a chunk request. A scheduler takes as input a set of parameters such as the buffer size, target buffer level [3], and the end-time of the previous download. Its output is the time at which the next request to the server is dispatched.

Scheduling of a chunk requests can either be *sequential* or *parallel* [93]. A sequential scheduler requests chunks one at a time. It is worth noting that this does not in any way imply that the request must be made immediately after receiving a response from a server. In contrast, there can be an inactivity interval between subsequent requests. A parallel, scheduler dispatches multiple chunk requests at the same time. However, this does not necessarily imply that each request is for a separate chunk. Because in some cases multiple requests are targeted at the same chunks with each request targeting a sub-segment. A parallel scheduler is mainly used when a client intends to use multiple interfaces or wants to access content from multiple locations.

2.4.1 Sequential Scheduling

In this thesis, we call the most basic of all sequential scheduling techniques used in HAS *progressive dispatch*. When using a progressive dispatch, a client sends a request, for the next available chunk, as soon as it receives a response, this aggressively ramps up a buffer. During the start-up period or when the buffer level is below a particular target (i.e. when a player is at the buffering-state), the progressive dispatch is the most appropriate scheduling mode. Commercial players such as MSS, Netflix [26] and YouTube [62], as well as some non-commercial players (e.g., DAVVI [94], FESTIVE [19] and Adobe OSMF [26]) are all known to use progressive dispatch to ramp-up their buffer. Furthermore, it is argued in [95] that for a client to get its fair share of the available bandwidth, progressive dispatch should be used at all times before the download of the maximum available video quality level. Because by removing the OFF period a client can better estimate its fair share of the available capacity.

[3]Note this depends on the implementation.

When the network conditions fluctuate, a rate adaptation logic is used to match every request to the available resource. Hence, it is highly desirable for the scheduling logic to be flexible. *Periodic dispatch* is such a flexible approach, it sends a request to a server in a specified time interval after receiving a response. In addition, periodic dispatch can be used to avoid buffer overflow [14], or to save energy when streaming in mobile environments [70, 96]. This effectively results in an ON and OFF traffic patten. Most implementations derive the time interval between requests (i.e. the duration of the OFF period) from the chunk size [26, 62].

Kupka et al. [27] have investigated the performance of periodic dispatch. They found a reduced performance with respect to TCP throughput, which, they observed, can be remedied through the use of large video segments. Akhshabi et al. [13] look at the behaviour of periodic dispatch scheduling when multiple players are trying to share a bottleneck link, and observe that there are issues concerning instability, unfairness, and under-utilisation of resources. Villa and Heegaard in [97] identified three likely solutions to both the unfairness and the oscillatory nature of the above switching logic. The first approach is to randomise chunks' inter-arrival time, and secondly, a back-off period is introduced.

A different approach is taken in [98, 99, 100], they advocate for a server push strategy. HTTP 2.0 allows a server to push content to a receiver directly without the need of an explicit request. Wei and Swaminathan [99, 101] presented three push strategies that exploit the server push feature of the HTTP 2.0, namely *no-push*, *pull-push* and *K-push*. In pull-push, after the initial request, the server sequentially sends chunks to a client without any break, and stop only when explicitly asked by the client. While in the k-push a client initiates a request for a block of k chunks after receiving the initial request, and then the server responds by sending then back-to-back except where the client terminates the request. Obviously, no-push is when no server push is allowed. The push strategies were found to reduce live latency and improve link utilisation [99, 101].

2.4.2 Parallel Scheduling

Parallel scheduling is required when a client has multiple network interfaces and uses some or all the interfaces simultaneously for a particular streaming session. Another case where parallel scheduling is required is when a client is streaming from multiple servers using one or more connections. Without appropriate scheduling, using multiple network interfaces does not necessarily guarantee a high-quality streaming service. In fact, poor performance is the expectation [102].

Kasper et al. [103] study the case where HTTP range retrieval is used to request chunks by a multi-homed client sequentially. The performance of the technique is found to be dependent on segment size. The authors proposed two possible solutions, i.e. either to get an optimal segment size or to parallelise the scheduling. In their follow-up work [104] they investigate parallelising the schedule since finding an optimal segment size imposes a trade-off between the throughput and the start-up latency. The proposed technique is based on an HTTP pipeline and allows a client to send a request without the need to wait for a response.

In contrast to approaches using HTTP range retrieval Liu et al. [105] propose a scheduling scheme that enables a client to request multiple chunks in parallel using independent HTTP sessions. The client-side scheduler first sends an HTTP GET request to a server, while the client is still receiving the requested chunk, it dispatches another request. Each chunk received is indexed. And the index is appropriately updated whenever a new chunk arrives or when a chunk download is finished. The system must not download more than a predefined upper limit of allowable parallel threads. Before requesting a chunk, the scheduler calculates the ratio of the duration of a sub-segment downloaded and the duration of the whole chunk. This is done for each of the chunks that are currently being downloaded in parallel. It then compares the ratio to a threshold value (for detail on how to get the threshold see [93, 105]). If the ratio of all the parallel HTTP threads is larger than the threshold and the buffer level is less than the upper limit a new chunk is requested in parallel. However, no new request is dispatched in case one of the following three conditions holds: (i) the calculated ratio is less than threshold value in at least one of the parallel downloads; (ii) the maximum allowable parallel sessions have been reached; and (iii) buffer level is equal or greater than the set upper bound.

2.5 Adaptation Function

The *adaptation function* is the element within the ABR scheme that decides the profile of a chunk to be requested (also called *adaptation logic* or *switching logic*). It usually takes information regarding the available capacity, the buffer level, the schedule of the next chunk, and the set of all the possible representations as its inputs. Then returns a particular representation of a chunk to be downloaded. However, when a server presents to a client video representations that the client is not capable of supporting, the adaptation logic should not consider them. For example, if a client does not support HD any video rate approaching HD resolution should be disregarded.

In its most elementary form, the adaptation logic just chooses a chunk with the highest video rate the estimated available resource can accommodate. This basic algorithm is hardly in use nowadays (even though it is simple to implement). Because it is susceptible to the time-varying nature of the resources that ABR schemes commonly rely on, which makes the outcome of this basic scheme oscillatory, abrupt [13, 62] and unfair in allocating bandwidth to competing clients [20]. Next are some of the most prominent techniques used by various researchers to realise a typical adaptation logic.

2.5.1 Heuristic Based Adaptation

Most of the early ABR schemes are based on heuristics. Liu et al. [18] heuristically implemented an AIMD-like adaptation logic that employed a step-wise switch-up and aggressive switch-down logic. They argued that this technique prevents video rebuffering that might happen if the switch-up is aggressive. And with an aggressive switch-down buffer is speedily filled up. However, the algorithm was found to intermittently chose sub-optimal representation and is unstable as it oscillates between different video quality levels.

Another player with a heuristic based adaptation module is FESTIVE [19]. It gradually switches to the highest video level with the rate of switching decreasing as the video rate of the chunk increases. It is assumed that this will mitigate the unnecessary oscillation between different video representations. Additionally, the paper introduces a notion of delayed update, which is a score that measures a trade-off between efficiency/fairness and stability, and allows a player to improve its stability. Mok et al. [29] studied the effect of video quality transition on QoE. They report that a sudden drop in the video rate has a negative impact on the user experience. To improve the QoE they propose a heuristically designed ABR called QDASH-qoe (a QoE-aware DASH system). The scheme switches down the video rate to an intermediate level even when the target video rate is lower. Although this may result in a sub-optimal choice, by improving stability, they are able to enhance the subjective user experience. Experiments by Akhshabi et al. [26] found that the MSS service is using a somewhat similar approach. Though for MSS the upward transition is faster than its downward trajectory, in either case, the switching is not immediate. Other players that employ a heuristic based adaptation logic are the player proposed in [22], AdapTech Streaming [66], and the Akamai HD Video Streaming services [106].

2.5.2 Control Theory Based Adaptation

It is hard to use heuristics to design an algorithm that is predictable and mathematically describable. Thus, recently there are some attempts to design an adaptation logic that is not only performing well but also based on descriptive and predictable models. Control theory is used to model dynamical systems that are stable, accurate and settle quickly into a steady state [107].

The work of Cicco et al. [108] proposes an adaptation logic based on feedback control. The quality adaptation controller takes a target buffer as an input and returns the video rate of the chunk to be downloaded. The goal of the controller is to ensure the buffer is always maintained at the target level. It achieves this by calculating the error between the target buffer and the measured buffer level. The error is then passed to the Proportional Integral (PI) controller that outputs a video rate that matches the estimated available bandwidth. However, since HAS works with discrete quality levels, the value is passed to a quantizer, which returns the highest representation that is less than the output of the PI controller.

Tain and Liu [21] propose a control theoretic client side rate adaptation that makes a trade-off between the stability in video rate and bandwidth utilisation. In [109] model predictive control (MPC) is employed to predict the expected throughput of the next couple of chunks, and then combine it with the buffer state information in making a decision on which bitrate is the most optimal in maximising QoE. Zhou et al. propose adaptation functions that are implemented using control theory, and others [68, 110, 111, 112].

2.5.3 Optimisation Based Adaptation

Qiu et al. in [75] tried a different approach. They use an optimisation technique for rate adaptation algorithm (called Intelligent Bitrate Switching based Adaptive Video Streaming). They model the adaptation logic as an optimisation problem, which maximises benefit and minimises penalty. The benefit is represented by the quality level of a chunk, with higher video rates having a higher benefit value. However, a maximum penalty is assigned to a video freeze. Interestingly, the user can adjust the penalty based on his viewing desires. The algorithm can also use subjective metrics like PSNR to assess the QoE. An optimal solution is expected to select a chunk with the highest video rate among all the chunks that satisfy the given the constrain of a minimum number of rebuffers. Another adaptation logic based on optimisation techniques is presented in [12].

At the network provider side, there is a need to arbitrate the allocation of network resources among the competing clients. Hence, according to Bouten et al. [113, 114] the support for coordinated management, and global optimisation is essential. They employ an integer linear programming (ILP) model to manage policies to either maximise the QoE of all users or minimise the penalties incurred for violating the subscriptions contract. Joseph and Veciana [115] propose NOVA to solve the multi-user joint resource allocation and quality adaptation problem employing optimisation techniques. The algorithm attempts to maximise the average video quality and minimise the quality variability of HAS streaming session subject to network constraints.

2.5.4 Artificial Intelligence Based Adaptation

Xiong et al. [116] argue that since a change in video rate affects the user's subjective perception of the overall streaming quality and the user perceived quality is not easily described in precise language, control theory and other mathematical models that rely on a precise definition of input and output are not necessarily the best options for implementing an adaptation logic. They propose an adaptation module based on fuzzy logic, called *Network-Bandwidth-Aware Streaming Version Switcher*. Vergados et al. in [117] also employ fuzzy logic to adapt the video rate to the changing network conditions, but unlike [116] their fuzzy controller uses buffer state changes as input. The aim of the algorithm is to prevent buffer overflow and unnecessary fluctuation in the video quality. However, the algorithm suffers from a high amplitude variation in video quality changes. To remedy this shortcoming the authors of [118] propose an AIMD-like fuzzy controller that takes into account both the estimated throughput and buffer occupancy and returns the appropriate video rate to be requested.

Using fuzzy logic requires the use of domain expert knowledge, which is difficult to get. Even where it is available, it is hard to define a set of linguistic rules [119]. Another artificial intelligence technique used in the video rate adaptation that requires no expert input is machine learning (ML) [120, 121, 122, 123]. With ML techniques a client can learn to adapt its video quality to the changing context without the need for any human intervention. Chein et al. in [122] use a decision tree based random forest classification to map network related features onto the video rate. The scheme trains the classification model using a dataset provided in [124]. The classifier is then used to predict the current request or any future video request. The training can be done either online or offline.

Classification schemes require a training dataset. However, in a highly dynamical system like HAS, it is difficult to get a training set that is both correct and representative for all possible situations. Reinforcement Learning (RL) allows an agent to discover the right action to take, within a particular context, based on feedback from its environment. To do this, an adaptation module interacts with its environment by sensing factors that are expected to influence its decision. For example, in [125] the RL agent senses average and the mean absolute difference in bandwidth, which the scheme in [120, 121] senses buffer state changes and available bandwidth. After sensing its environment, the RL agent takes action, typically changing the video rate to incrementally maximise its reward (such as improving the Mean Opinion score and reducing the rebuffering) [125].

2.6 Interaction of Components

As previously discussed, the ABR module, like most non-trivial systems, is composed of several components, grouped into three elements. These subsystems interact with one another and their operating environment. In Figure 2.5, an overview of these interactions is presented.

The relationship and interaction between the throughput estimation subsystem and the buffer management function have been receiving the highest attention [62, 26, 20]. It has been found that the greater the available network capacity, the faster the chunk download [21]; therefore the faster the replenishment rate of the buffer. However, many algorithms such [22, 76], try to maintain buffer at a certain level. For this, the scheduling function activates the period dispatch, which results in an ON-OFF traffic. As discussed in Section 2.3.1.4, this traffic pattern results in the drop of TCP throughput that in turn reduces the speed of the buffer replenishment. Naturally, this creates a cyclic relationship that goes in an opposite direction. To ensure that the client perceives the accurate available bandwidth, the scheduling process has to take this into account when deciding the time of the next request. In [14, 23, 25] it is recommended that the periodic dispatch is only activated when a buffer is full or nearly full. In other words, while the ABR scheme is ramping-up its

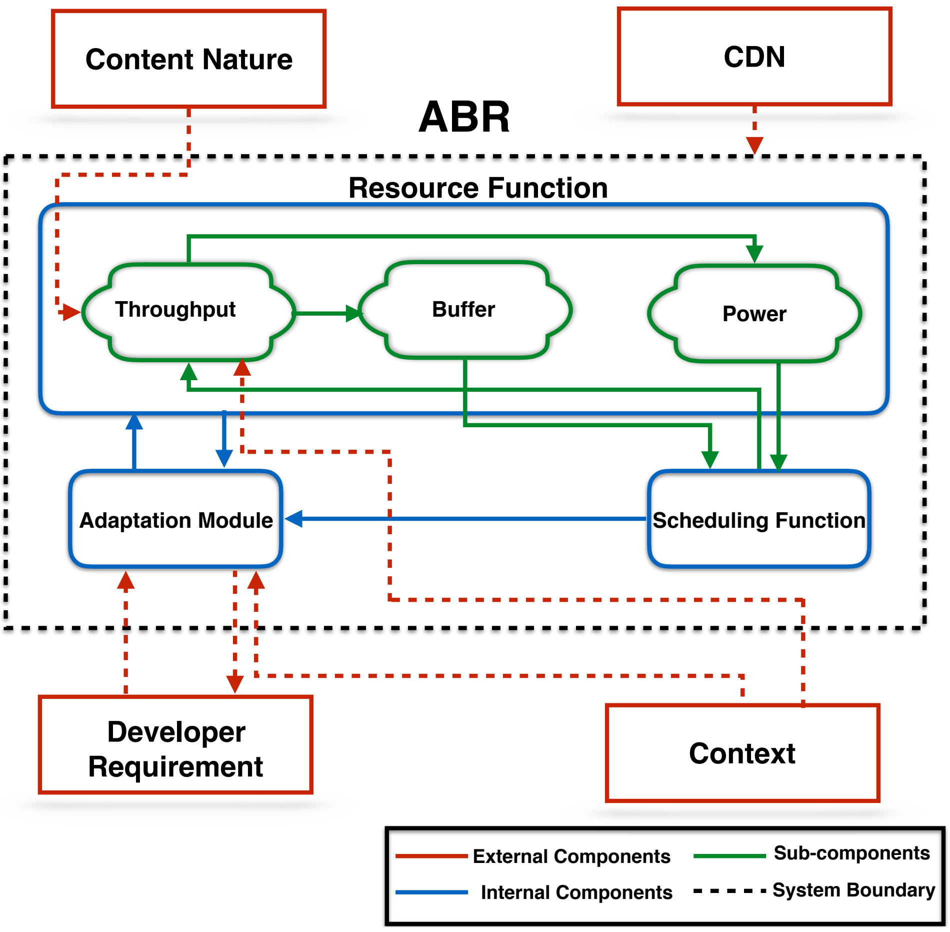


Figure 2.5: Interaction of ABR Components.

video rate only the progress dispatch is to be used. The authors of the papers argue that this will ensure that provided the highest video rate has not yet been reached, the OFF period is not activated.

However, the buffer management subsystem is not the only component interacting with the resource estimation function; others are the streaming context monitor and the user requirement sub-system. Usually, these are outside the control of most ABR designers. For example, an ABR algorithm developer has no control over the impact of weather (humidity) on the wireless channels.

The adaptation Module takes the output of both the resource estimation function and the scheduling modules as its input and then decides on the next video rate to request, subject to the designer's constraints and policies. For example, An ABR designer may want the video oscillation not to exceed a certain threshold [21], or the start-up delay must be within a particular range [65, 66]. Therefore, the adaptation logic must ensure that the selected video rate does not result in a violation of any of the constraints. It should be noted that some of these constraints may evolve with time. In this situation, the current action of the adaptation module decides on the next step. Hence interaction becomes two-way.

Another thing to note is that while an adaptation module does not directly affect the scheduling function, it does interact with it through its interactions with the various subsystems of the resource estimation function. Which video rate is requested has an impact on all the subcomponents of the resource estimation function discussed. The most obvious of these interactions are between the buffer replenishment rate and the video rate of the requested chunk since chucks with a lower video rate contains less data and vice versa. Hence, assuming the available bandwidth is constant, the buffer replenishment rate has an inverse relationship with the video rate. However, the throughput perceived has a proportional relationship with the video rate. As discussed in Section 2.3.1, there is a feedback loop between the data downloaded and the TCP mechanism, with a longer download allowing TCP to have a better chance of reaching a steady state. The next sections will discuss the impact of external factors.

2.7 Context Management

Until recently, video quality adaptation decisions have been mostly based on the system level factors discussed at the Section 2.3. However, it is becoming increasingly clear that to ensure a high level of user satisfaction; system level information needs to be complemented by context-dependent information [126, 127, 128, 129, 130]. Context is defined as ‘any information that assists in determining a situation(s) related to a user, network or device’ [126]. As observed in [127] the first requirement towards exploiting context, in improving user experience, is to identify the set of parameters that together defined a specific context, such that a change in any parameter results in a change of context. Though ‘context’ is a very complex and loaded term, it is obvious that the more parameters used in defining a context, the better. However, this will be difficult to manage and model. Therefore, to better manage this complexity the authors of [126] grouped these parameters into four dimensions:

1. Device related parameters e.g. screen size, layout;
2. User and environmental parameters e.g. location, weather;
3. Application based parameters e.g. type;
4. Network level parameters e.g. throughput, packet loss, RTT.

A context vector, $CV = \{1...n\}$, is a set of combination parameters from the above groups, which uniquely identify a context. When the CV consists of parameters from one group e.g. device related parameters, we call it a *uni-group* context; otherwise it is called *multi-group* context. A uni-group context relying on network level parameters has received most of the attention of the HAS community (see Section 2.3 for detail). But this is gradually changing. The first generation of multi-group context-based ABR schemes, use network level parameters as the main factors, while other factors are used either as adjustment factors or for improving the accuracy of the network level parameters.

2.7.1 Context as Complementary Parameter

In a wireless environment, the channel capacity is inherently varying and difficult to estimate because of the variation in signal strength; interference from other devices; environment induced noise and user mobility. These context-induced impairments, which manifest as an increase in bit error rate, packet loss, and delay, complicate the task of TCP. Because TCP traditionally views all packet loss as a sign of congestion, and when this is not the case, an unnecessary reduction in end-to-end throughput and increased delays occur.

Another solution to this challenge, in addition to those outlined in Section 2.3.1, is to incorporate environmental parameters in the process of throughput prediction. The current location of a streaming device is the most prominent environmental parameter currently, but more factors are expected to be used in future. This is mostly because, as shown in [127, 131], location can be a better indicator of the actual link capacity when the characteristic of the streaming environment changes often. Several attempts have been made to improve the accuracy of TCP throughput estimation by incorporating location-based data [132, 133, 134, 135, 136].

In [134], a location-based bandwidth lookup service is proposed to help streaming clients to predict the available bandwidth better. The authors use video receivers equipped with GPS capturing capability to collect the bandwidth of popularly commute routes in Norway, which they used to build a bandwidth lookup database. A client streaming while commuting along the mapped route can query the database with its current location. The service responds with the near-future available bandwidth, which the client may use to adapt the video rate. In [136], a crowdsourcing technique is used to collect the location-bandwidth information from different devices running a variety of applications. The authors observed that this presents a challenge because frequently the participating devices are in an idle state. Hence, the data collected may not be the accurate estimate of the available link capacity. To improve the accuracy of the lookup data, for a given route interval, they multiply the size of each record of the downloaded data by the corresponding record of the data throughput and then summed up all the reading before dividing the result by the total size of downloaded data. They argue this gives more weight to the high throughput data.

In [132], it is streaming clients that send data about their geolocation and bandwidth estimates to a server, which are kept in a repository. When a new or an already participating client desires to stream a video, in a particular location, it first sends a query to the server. And the server will use the client's GPS to predicts the future path of the client, thereafter the server determines the possible bandwidth along the predicted path and sends it to the client.

In [137] it is argued that location is not the only contextual parameter that can be used to enhance the accuracy of throughput estimation. Another important parameter they suggested is the time of the day. This is since wireless channels are always shared, so the allocation of bandwidth will normally depend on the number of active connections, which varies at different times of the day. Relying on the geostatistical methods, they analyse the impact of both space and time on the bandwidth prediction. They then used the spatiotemporal model derived from using variogram to capture the relationship between distances among sample, time, and semivariance, together with an interpolation method to predict the future available bandwidth in an unknown location. Their result shows a more accurate estimate of the available bandwidth.

2.7.2 Context as an Adjustment Factor

So far, we have seen contextual information only being used to improve the accuracy of the estimated network capacity. In other words, the contextual information is not directly used by the adaptation logic in making a rate selection decision. Though an early stage of development, using contextual parameter directly as adjustment factor is increasingly becoming common. First, context is monitored and measured. Then the result, usually a multi-group context vector, is fed into the adaptation module.

In [135], CV={Network type, humidity, location, speed, time, throughput} is used as input the adaptation logic. The network interface can either be 4G LTE or 3G. The authors first investigate how these factors affect the available throughput of the two types of the network technologies. Their findings show that humidity and location are the dominant factors when streaming over 3G, while speed and time are more important factors when streaming over 4G. To request a video chunk, a client sends its current measured CV, the server then matches the client CV to an entry in its database, and retrieves bandwidth attribute of the tuple that matches the client's CV. Based on the current throughput in the client's CV and the retrieved bandwidth, the server determines the appropriate video rate to send.

In [127, 138], location information is used to find the 'close-to-optimal' buffering strategy that prevents video stalling in an area of limited connectivity. The adaptation logic adjusts its buffer based on how close it is from a coverage hole, such as a tunnel; and how long the hole is. The closer or the longer the tunnel, the more the buffer space that is allocated, and the lower the video rate of the requested chunks. However, for the scheme to work the authors assumes that a client has the ability to predict its path and knows where the areas of interest are located.

2.8 Quality Metrics and Measurement

The primary object of a typical ABR algorithm designer and developer is to ensure a high level of QoE, which is defined as "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" [139]. Two issues currently dominate the research actives in the field of QoE. First, is what are the factors that affect QoE, and secondly, in what way and manner [15].

Perceived video quality is usually measured using subjective tests. With this methodology, human subjects are shown video, usually in a controlled environment configured using different parameters, e.g. different video rate and varying length of video stalls [140, 141]. The data gathered is analysed using correlation analysis techniques [142] to deduce the relationship and the dependencies between different parameters. For example, in [143] the relationship between video rate and the join time is investigated, while in [1, 144] the relationship between video rate and structural similarity (SSIM) is investigated. Based on this type of analysis the following factors are found to affect QoE the most:

- Video freeze.
- Video quality fluctuation.
- Start-up delay.
- Average video rate.

When the buffer is completely depleted, either because of network failure, a request of video rate above the capacity of the transmission link, or any other reason, the player goes out of content and a playback stalls. This is called video rebuffering or freeze. Some research work has shown that both the frequency and the duration of video freeze event have an adverse impact on user experience [15, 140, 145]. It was found In [146] that user satisfaction decrease with increase in the duration of rebuffering event. In fact, the authors [147] found there is an increasing probability of user abandoning a video streaming session as an interruption period prolongs. Furthermore, it is shown in [148] that independent of the duration of the freezing events the frequency of video stalls has a significant negative impact on QoE. However, users are found to prefer one long video freeze to frequent short video stalls [146]. Therefore, ABR designers are advised to avoid video rebuffering at all cost [15].

As network condition changes, such as in wireless environments, ABR schemes typically try to change the video rate of their request. While this is good, as it allows an ABR algorithm to prevent video rebuffering, if the video rate switch happens frequently, the video rate will oscillate. A result of a subjective experiment conducted in [149] to access the impact of the frequency of video quality switches on user experience has shown an increasing deterioration in user experience as the number of quality changes increase. Furthermore, Liu et al. [142] have found the perceived degradation in the quality of video streaming session increases as the amplitude of a video rate switch increases. They also showed that the degree of the degradation in QoE depends on whether the video quality is being increased or decreased, with an 'increasing switch' having a much smaller impact than a 'decreasing switch'. However, Samira [140] has found that the impact of video rate switch decreases as the video rate increases.

Start-up delay is the amount of time a user has to wait before the first frame appears. Its size usually depends on the buffer size, and the video rate of the chunks being requested at the buffering phase [62, 15]. The various researches on the impact start-up delay on the QoE have, so far, been showing a mixed result. The authors of [147] have found that an increase in the start-up delay increases the probability of user abandoning a streaming session, however, other researchers [140, 150] found no significant increase in video quality degradation. In fact [148], 'increasing initial delay' to reduce the chances of a video stall has been recommended.

Overall, the average video rate is found to increase QoE [142, 15]. Nonetheless, it has been shown in [1, 28] the relationship the video rate and the perceived quality is non-linear. In other words, as the video rate increases a point is reached when the quality experience by a user saturates [1], which results in users not being keen on further increase of the video rate [28].

However, regression analysis, as is currently used in most of the QoE research, may not be able to capture all intricacies and subtleties of the relationship that exits between these QoE metrics. For instance, while an increase in the video rate increases the start-up delay, and increasing the startup delay may help in reducing number of video freeze, which incidentally has an inverse relationship to the video rate. In [151] a quasi-experimental design (QED) is used to analyse the causality between video quality and user engagement. The study reinforces some of the findings of the correlation analysis. For example, increasing eithier the startup delay or video stall increases the chances of viewers abandoning a video session. However, they also found that the impact depends on the combination of the network technology e.g. Wifi; streaming device e.g. TV or smart phone; and the type of the video service (live or VoD).

2.9 Summary

In the past five years, tremendous effort has been put into standardising and improving HTTP adaptive streaming (HAS). This chapter structures and summarises the related research in the area. After presenting the evolution of Internet video streaming up to the advent of HAS services, a review of adaptive bitrate selection (ABR) was presented, which is part of the HAS service that decides the profile and schedule of a chunk to be downloaded by a video streaming client.

The chapter then presented a framework that decomposes the ABR into three components, that is, resource estimation module, chunk scheduling function, and adaptation logic. Then a discussion of the benefits and challenges of locating, monitoring, and measuring the various resources (e.g. throughput) that different ABR schemes rely on for their decision-making was presented. After this comes a classification of the chunk-scheduling algorithms into sequential and parallel schedules, which is aimed at clarifying where and when it is most appropriate to employ either of them. Considering the fact that the technique used in implementing rate adaptation dictates the limit of achievable optimisation. Hence, the chapter presented a detailed discussion of the various approaches used by the state of the ABR algorithms (e.g. machine learning or control theory) in designing and implementing ABR. After that, a detailed discussion on how these components interact with each other was presented. The chapter concluded with a report on the impact of the external factors, mainly the operating context and the user requirements, on the performance of an ABR module.

From this extensive discussion, the following lessons can be derived:

- **Resource estimation:**
  - The choice of scheduling policy is more important in improving the perceived throughput than the throughput estimation technique.
  - A good ABR scheme requires at least two factors to work satisfactorily.
- **Scheduling function:**
  - Employing the progressive dispatch at the ramping-up stage makes it easier for the client's TCP throughput to converge.
- **Adaptation function:**
  - The technique used in realising the adaptation logic has no significant impact on the performance of an ABR scheme.
  - The simpler the adaptation logic, the better.
- **External factors:**
  - Contextual information is an excellent source of TCP throughput improvement.

– To perform effectively an adaptation logic requires the input of the evolving state of QoE metrics in addition to the conventional resource metrics.

However, there are still challenges that do require the attention of ABR researchers. First, there is a need for a clear articulation of the relationships and interactions between the various subcomponents of the ABR module. For example, when building a player, an ABR algorithm designer requires the exact relationship between video quality and the buffer state changes. Another area that requires further work is how to monitor, measure, and incorporate the various QoE metrics into video quality selection decision. QoE feedback mechanisms can help a client in ensuring that the requirements of the various stakeholder are met.

The next Chapter will present a framework that unifies the relationship between the buffer state changes, throughput, and the requested video rate. And then it will derive an analytical model which relies on the pattern of video quality changes that are known to affect user experience.

Chapter 3

Video Rate Evolution

3.1 Introduction

In the previous chapter, a comprehensive review of the HTTP adaptive video streaming is presented. It was observed that the relationships and the interactions between the various components of the ABR still require further elaboration. Since not all interactions will result in the improvement of the user experience, to ensure that only the interactions that help improve QoE are derived, we have to, first, carefully define the valid states that the system can be in, at any given time. However, states in isolation may not tell us much, because the user experience is not an isolated event, but rather evolves. Therefore, a state evolution rule that dictates the valid state transitions from any given state is required. Only after constraining the ABR system can we be sure that the relationship and interaction derived are the desirable ones.

The chapter begins by defining the vector that constitutes a valid state. Then we identify the patterns of state changes that affect the QoE. From this, we present a state transition map that defines the only valid trajectory across which the system state shall evolve. The proposed model is declarative, hence only allows us to understand how the system changes. To demonstrate how an ABR algorithm designer can leverage this information, a throughput-based and a buffer-based ABR schemes are modified to work with the proposed model. And finally, the results of experiments used to evaluate the modified schemes are presented.

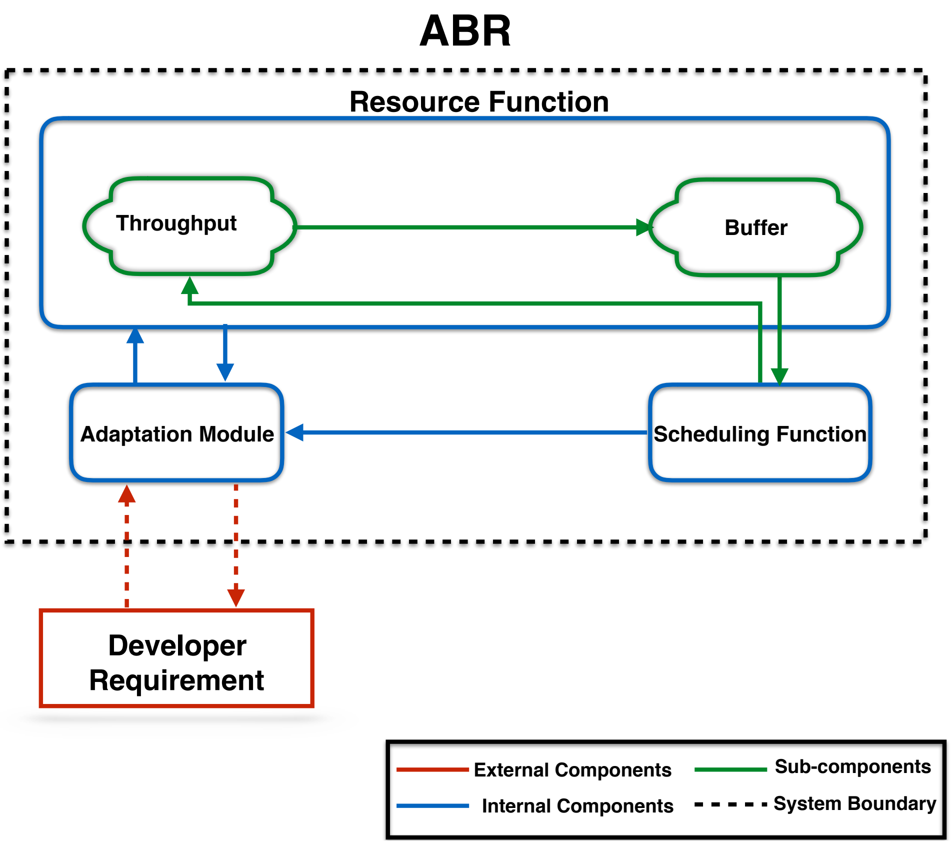


Figure 3.1: The activity diagram of the Classic Framework

3.2 System State Space

As seen in Chapter 2, like most non-trivial systems, an ABR module is composed of several internal and external components, with these elements interacting complexly. Any of these interactions may result in a change of the state of an ABR module. To reduce the complexity of the system design, in this chapter, we assume that an ABR algorithm designers have complete control over the internal components of the ABR system, and have little or no influence over the external subsystem. Figure 3.1 presents the activity diagram of the internal components of an ABR module, interacting with one another to ensure that the user requirements are met, henceforth called the *Classic Framework*. The framework is derived from Figure 2.6, with all external influences, omitted.

As can be observed, in the Classic Framework, there is only a one-way direct interaction between the throughput estimation module and the buffer management function. In this relationship, the throughput estimation function is the independent component while the buffer management subsystem is the dependent component. In other words, as the value of the estimated throughput changes the effect on the buffer occupancy is observed and noted. But recall, the chunk request scheduling process, which has a significant impact on the perceived throughput, is dependent on buffer state changes. This creates an indirect interaction, in the opposite direction, from the buffer management module to the throughput estimation function via the scheduling function resulting in a relationship that is cyclic and anisotropic.

In a first step, it is necessary that the described loop is broken. Doing this will ensure that the current state is not changed by forces that are difficult to control. We can achieve this by making sure that when making a scheduling decision throughput and buffer state changes are jointly considered. A simple scheduling policy that can allow us to reach this goal is guaranteeing that the periodic dispatch (ON-OFF) scheduling is only used when required, that is, only when preventing buffer overflow. Policy 1 summaries this, as:

Policy 1 *: The progressive dispatch scheduling should be used, except when the buffer is full.*

This policy makes it easy for the TCP to reach the steady state, and where it does not, we will be in no doubt that, this is not as the result of the interference from the scheduling function. This modification of the Classic Framework results is a new set of relationships and interactions among the ABR components, henceforth to be called the *Framework 1*, presented in Figure 3.2.

From Framework 1, we can deduce that the state of an ABR scheme ($\mathcal{S}$) can be defined by the current throughput and buffer occupancy:

Definition 3.1

$$\mathcal{S} = (throughput, buffer)$$

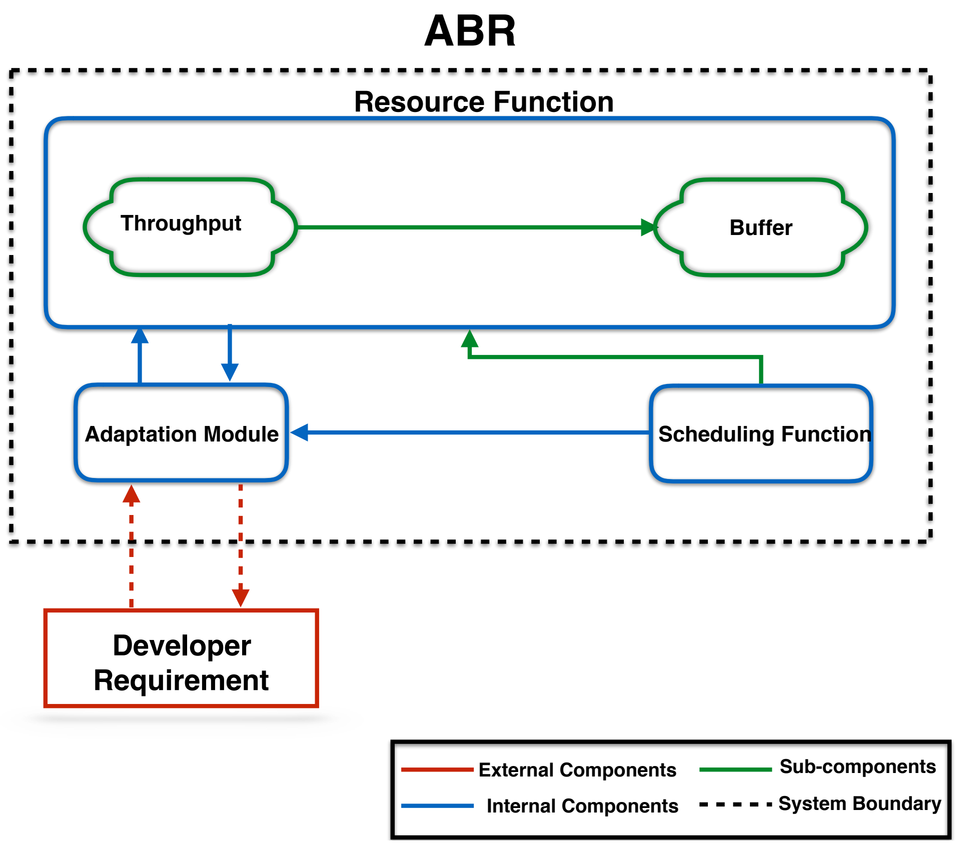


Figure 3.2: The activity diagram of framework 1

Using Framework 1, the adaptation logic, just like in the Classic Framework, takes the current $\mathcal{S}_i$ as input. It then uses the scheduling scheme outlined in Policy 1 to decide on the video rate that meets the developer's set of requirements. This chapter assumes that an ABR algorithm designer is interested in a service that guarantees the following set of QoE requirements:

- high average video rate,
- low video rate oscillation,
- low startup delay,
- high network utilisation,
- lowest possible number of rebuffering events.

Let us assume that a rate selection function $R()$ represents the adaptation logic, this will then shield us from implementation details, which as we have seen at Section 2.5 may vary depending the technique used. Formally, this can be defined as:

Definition 3.2

$$R(\mathcal{S}_\rangle) \rightarrow (\mathcal{S}_{i+1}, video\,rate)$$

The Definition 3.2 states that given the current state $\mathcal{S}_i$, $R()$ will return a video rate to be requested and results in a new state $\mathcal{S}_{i+1}$ that can be the same with or different from the previous state $\mathcal{S}_i$. In other words, invoking $R()$ can result in a change of either the current throughput estimate or the buffer occupancy or both. Therefore, the adaptation must ensure that any video rate return must not result in a situation that adversely affects any of the developer's requirements. However, for this to happen $R()$ needs a mapping between the two parameters and the video are, that is, $(throughput, buffer) \rightarrow video\,rate$.

The mapping between throughput to video rate, $(throughput,) \rightarrow video\,rate$, is usually governed by the following simple policy:

Policy 2

$$video\,rate \leqslant throughput.$$

Policy 2 simply states us that on no account should the $R()$ select a video rate that is more than the currently estimated throughput. It does not specify the video rate to request at any given time, but rather defines a constraint within which the system can operate. Any action outside this boundary will result in an invalid state [1].

Subsequently, the mapping between buffer level and the video rate, $(, buffer) \rightarrow video\,rate$, has to be defined. Since $R()$ cannot rely on throughput to decide on the video rate to select, it will have to be based on the buffer level for rate adaptation decisions chiefly. The definition of the system state $\mathcal{S}$ s further restricted as thus:

[1]While in theory; it is possible to use throughput as a decision variable in predicting the video rate of a chunk to be requested, as discussed in Chapter 2, this may result in frequent video rate oscillations or under-utilisation of the available resource. Hence, throughout this thesis, we use throughput only either as a constraint, or an adjustment factor.

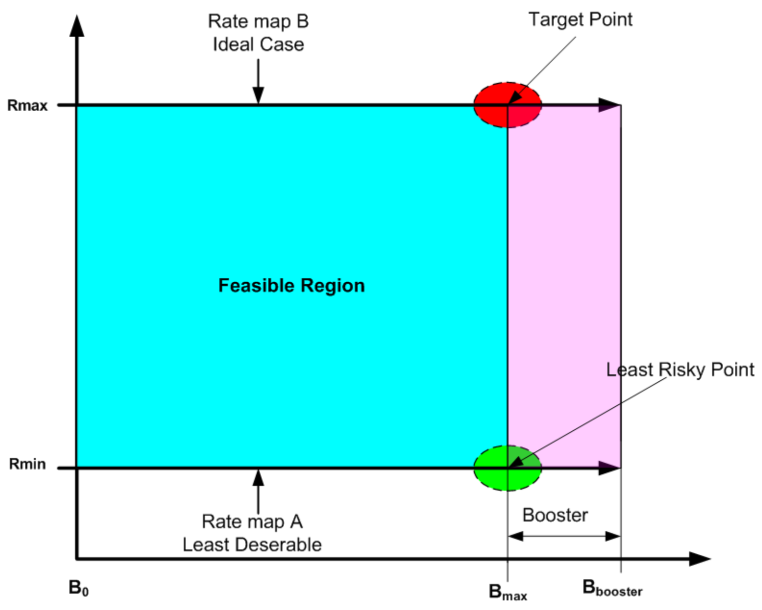


Figure 3.3: All the possible mappings between buffer level to video rate

Definition 3.3

$$\mathcal{S} = (buffer)$$

$$subject\ to\quad throughput - video\,rate \geq 0.$$

Let us assume that the maximum video rate is R_{max} and minimum video rate is R_{min}, therefore for any given buffer level $R()$ has to decide on which video rate to select. Figure 3.3 presents a plot of all possible $(, buffer) \rightarrow video\,rate$ combinations. The vertical axis represents the video rates from the minimum to the maximum, and the horizontal axis shows all the possible buffer levels. With B_{max} being the maximum buffer size allowable for rate selection. However, it should be noted that the buffer serves other purposes, as well. For example, some portion of the buffer can be dedicated to absorbing short-term network outages, hence prolonging the viewing session even in the absence of incoming content. In this thesis, we call the part of buffer reserved for any reason other than video rate selection: *Buffer Booster*, which is coloured pink in Figure 3.3. This chapter will not consider the use of the buffer booster in any detail.

The feasible region, coloured blue, indicates the only area within which an ABR scheme can choose a video rate given a buffer position. Within this region, for any buffer level, an $R()$ is constrained to select only from the range of the available video rates specified in the MPD. Therefore, from when the buffer is empty to the point it is full, the system can be in any of these $|Q| * (B_{max} + 1)$ number of states. However, in practice, a state is not typically chosen in isolation but rather based on the previous states of the player. This leads us to the next important concept: the video rate evolution trajectory.

3.3 State Evolution Trajectory

When we go back to Figure 3.3, it is easy to see why we are not interested in all states. As c can be observed, a streaming session typically starts by an ABR scheme requesting the lowest video rate, because the buffer at the time is empty, with the ultimate aim of ramping-up its video rate to the maximum available by the time the buffer is filled up, at the latest. A series of state traversals that move the system from the initial state to the target state defines the system trajectory, henceforth called the *rate map*. Always, rate selection decision will be restricted to these 'valid' states'.

Even when the choice is limited to a rate map, there is still a significant number of alternatives to choose from. Fortunately, many of these options are intuitively sub-optimal. For instance, looking at the same Figure again, (Figure 3.3) Rate Map A shows a scenario whereby given any buffer level $R()$ selects the minimum video rate. Even though this rate map is likely to result in the least number of video stalling events, it will naturally subject a user to the lowest video quality. Another example is Rate Map B, with which the $R()$ can only choose the highest video rate, this rate map will certainly prolong the start-up delay and increase the risk of video freeze.

In [14, 23], a rate map that divides the buffer into two parts is proposed. In the first part, given any buffer level the $R()$ selects a chunk with the minimum video rate, and after that, the video rate selection function linearly increase the video rate as the buffer level raises until the maximum bitrate is reached. However, by separating the buffering phase from the steady-state phase, a disconnected flow is created. Furthermore, at the phase when only lowest available video rate is downloaded, there will be a loss in video quality, and the longer this period takes, the more the user experience suffers. Furthermore, it has been shown in [91] that the probability of buffer depletion decreases exponentially with respect to the initial buffer level. Therefore, a linear increase of the video rate, when ramping-up, will unnecessarily prolong the convergence time. Also, it is worth recalling from Section 2.8 that the relationship between video rate and the quality perceived by a user is nonlinear. In fact, the authors of [28] have shown that when video quality is high, a rise in the current video rate does not necessarily translate into an equivalent improvement in the user-perceived quality.

To ensure that the video rate evolves in a way that optimises QoE, there is a need for a *rate evolution map* that captures the desirable pattern of video quality transition. Next sections will concentrate on the doing this.

3.3.1 Modelling Video Rate Evolution

At any given time, t after the video streaming has started the buffer may contain an array of chunks of different quality levels. However, segments of different video rates have different sizes in bytes. The thesis assumes that all chunks contain an equal amount of video time V in seconds. Since there is no direct mapping between buffer size in bytes and video time, the buffer is calibrated in a time unit, i.e. in seconds (see Section 2.3.2.1 for more detail).

At the beginning of a streaming session ($t = 0$), a server presents to a client a set of different video rates $Q = \{q_0, q_1, q_2 ... q_n\}$, with $|Q| = n + 1$. Let us suppose $q_0 < q_1 < ... < q_n$, therefore q_0 is the minimum quality level (referred here also as q_{min}) and q_n is the maximum available quality level (called q_{max}). Suppose B_t is the buffer level at time t and B_{max} is the maximum buffer. Let $\hat{c}_t$ denote the estimated throughput at time t with $C(t)$ being the system capacity (i.e., $\hat{c}_t \leq C_t$).

Usually, after receiving the MPD file at t_0, the play-out buffer is empty ($B_{t_0} = 0$). As seen in previous section, a client starts requesting a chunk with q_{min} in order to minimise the start-up period. However, a prolonged download of q_{min} will negatively affect the user experience. Hence, using $R()$, the client should immediately start a gradual improvement of the bitrate of the requested chunks as soon as it receives the initial chunk, such that the video rate of chunk $i + 1$ requested after successful download of chunk $i \geqslant 1$ with video rate q_k , where $\{k : \ 0 \leq k \leq n\}$, is $q_{k+1} = \alpha q_k$, where $\{\alpha : \ 0 < \alpha \leq 1\}$. Suppose that the download of chunk $i + 1$ with starts at time t^s and finishes at t^e. Let us also assume that the rate at which the client's requested video rate evolves with respect to time $dR()/dt$ is $g\prime(R)$. Assuming that $C(t) \geqslant q_{max}$, so that Policy 2 is not violated throughout. This makes $g\prime(R)$ positive at any time after the start of streaming except when $R() = q_{max}$ (because the system cannot request video rate higher than the maxim), and when $B_t = 0$, in which case $g\prime(R) = 0$.

To avoid high amplitude variations (e.g. an abrupt drop of the video quality), which are known to be detrimental to QoE [29, 152], transition decision to q_{k+1} should depend on q_k. Furthermore, since users are not known to be appreciative of an increase in the video quality when the video rate is relatively high [28] it is recommended that a non-linear $g\prime(R)$ is used. In fact, Yamagishi and Hayashi [145] have shown that the saturation with the increase in quality begins to take effect about half-way through the available video rates. One way to achieve this nonlinear increase in video rate is to continuously adjust the value of q_{k+1} with the distance between the current video rate and the maximum rate; this will ensure that the lower the selected video bitrate, the faster the rate of the rise in the video rate and vice versa. The trajectory of $g\prime(R)$ is concave path pinned at two points $q = 0$ and $q = q_{max}$ with amplitude at $q_{max}/2$. This pattern can be described easily by a quadratic function with $q = 0$ and $q = q_{max}$ and a positive constant α. It should be noted that the value of the constant α determines the vertex of the parabola. In other words, it determines the maximum value the $g\prime(R)$ can attain

$$g\prime(R) = aq(q_{max} - q) \tag{3.1}$$

3.3.2 Modelling the Video Rate Map

In the previous section, we have seen when to be fast and when not be with video rate increase while an ABR scheme moves from the lowest video rate to the highest.

In this section, we map this transition pattern to the available buffer space. But first recall from Definition 3.3 that for a state to be valid, the video rate requested that results in it must not be greater than the available capacity. Therefore, before anything can be done, we have to have a means of estimating the available network capacity.

Without loss of generality, we model $R()$ as a continuous function [2], that is, $R()$ can return in any value between q_{min} to q_{max}. Recall from Section 2.3 that clients usually infer $C(t)$ (the actual available bandwidth) from $c(t_i)$ (the estimated TCP throughput) for the purpose of rate selection. Now let us suppose that $c(t_i)$ is derived from the average of h number of previously observed $\hat{c}(t)$ (per-chunk throughput), and calculated as thus:

$$c(t_i) = \frac{1}{t_i - t_{i-h}} \int_{i-h}^{h} \hat{c}(t)dx. \tag{3.2}$$

In compliance with the Policy 1, a HAS client requests chunk i immediately after completely downloading chunk $i-1$ except when the buffer is full. In which case an ABR scheme waits for at least V seconds (chunk size) before sending a request. With this scheduling policy, except during the off period, the playback buffer drains at the one buffer second every real-time second and fills at $c(t)/R()$, therefore the rate at which buffer changes is

$$\frac{dB(t)}{dt} = \frac{c(t_i)}{R()} - 1. \tag{3.3}$$

In most contexts, especially the wireless environment, $c(t)$ is time-varying. Therefore, if the video rate selection is not to violate the Policy 2 and avoids buffer starvation, the output of $R()$ has to adapt to the changing environment as time passes, but recall, any such adaptation has to be as dictated by the Equation 3.1

$$\frac{dR()}{dt} = \alpha q(q_{max} - q). \tag{3.4}$$

[2] In the next section, we will drop this assumption and show how the proposed model can work with discrete video rate.

However, tracking the rate of change of the video rate with respect to time has two problems. First, it implicitly assumes an infinite buffer size, when it is not. Secondly, our interest is on the $(buffer) \rightarrow video\, rate$. To derive the rate of video change with respect to buffer changes, the Chain Rule is used.

$$\frac{dR()}{dt} = \frac{dR()}{dB}.\frac{dB}{dt}. \tag{3.5}$$

It can be observed that the right hand is composed of the Equation 3.3 and the what we want, that is the rate at with $R()$ changes its output with respect to buffer changes. Since $R() \leqslant c(t)$, as the output of $R()$ of approaches q_{max}, the rate at which buffer changes will tend to zero, that is, $\frac{dB(t)}{dt} = 0$. In other word, when $R() = q_{max}$ when $B_t \rightarrow B_{max}$. Hence, we have

$$\frac{dR()}{dB} \approx \alpha q(q_{max} - q) \tag{3.6}$$

$$\frac{dR()}{dB}\frac{1}{q(q_{max} - q)} = \alpha$$

after simplification using partial fraction method and using

$R() = q$ we have

$$\int \frac{1}{q} dq + \int \frac{1}{q_{max} - q} dq = \int \alpha q_{max} dB \tag{3.7}$$

by integrating equation (3.7) we have

$$\ln q - \ln |q_{max} - q| = \alpha q_{max} B + e \tag{3.8}$$

Recall, the streaming starts with a minimum quality level, therefore $q = q_{min}$ and $B = B_{t_0}$. Using this information e can be evaluated as thus.

$$e = \ln \frac{q_{min}}{q_{max} - q_{min}} - \alpha q_{max} B_{t_0} \tag{3.9}$$

Substituting equation (3.9) into (3.8) and simplifying results in

$$\ln\left[\frac{q}{q_{max}-q}\right] - \ln\left[\frac{q_{min}}{q_{max}-q_{min}}\right] = \alpha q_{max}(B_t - B_{t_0}), \tag{3.10}$$

finally solving for q and $(B_t - B_{t_0} \approx B_t)$, since $\{B_{t_0} : 0 < B_{t_0} \leq V\}$

$$R() = \frac{q_{max}}{1 + [\frac{q_{max}}{q_0} - 1]\mathrm{e}^{-\alpha q_{max} B_t}} \tag{3.11}$$

Equation 3.11 gives us the output equation, that is, the rate map. From any $\mathcal{S}_i$ this rate map should be able to take the system to next state $\mathcal{S}_{i+1}$, without violating any of the set policies. Please recall from the Definition 3.2, $R()$ should take the current state, and return a video rate, with this resulting in a new state that can be the same or different from the previous state. Equation 3.11 is in complete arrangement with the definition, because it return the next video rate by taking only the current buffer level as its input. In the process, resulting in a new buffer level that is or not equal to the previous buffer level. Next, we look at how does behaviour of the model guarantee the set of user requirement earlier set, but before that, the way in which the rate map can work with discrete video rates is presented.

3.3.3 Discrete Rate

By dropping the assumption of the continuous nature of video rates, the video bitrate has to be chosen from a finite discrete set. Therefore, $R()$ select from within this restricted set of video rates. As suggested earlier, video rate change is done only between adjacent video rates, that is, q_k can only move either to q_{k-1} or q_{k+1} to prevent high amplitude variation. Furthermore, the when the video rate return for a given buffer level is not an element of the valid set, the ABR scheme must disregard it.

The model is now modified to reflect this. To change a video rate, a buffer must have grown or contracted by a certain buffer distance. Precisely, to change the video rate $\Delta B_k = R^{-1}(q_{k+1}) - R^{-1}(q_k)$ is needed. In other words, all states that lie within

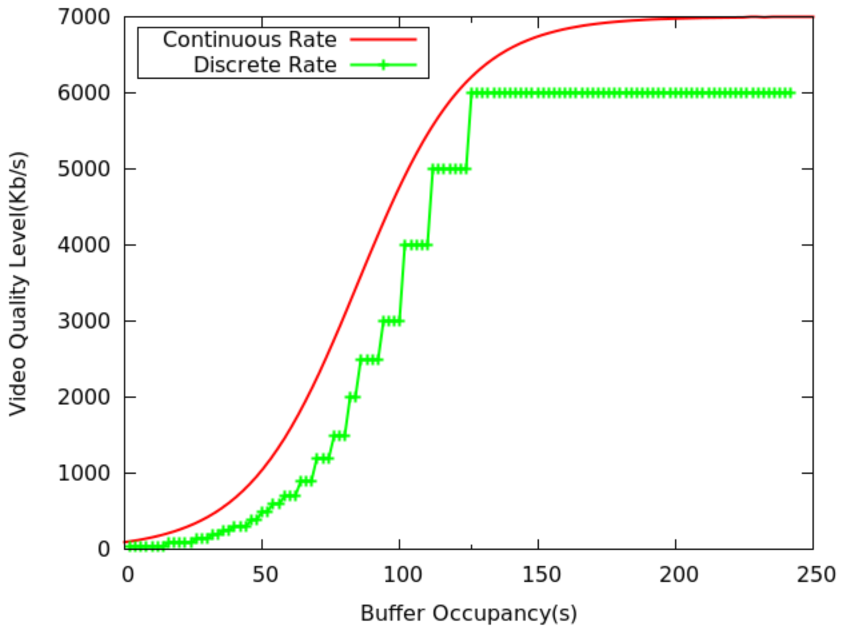


Figure 3.4: The evolution of the $R()$ in both Continuous and Discrete mode.

ΔB_k are treated as the same state by $R()$ for video rate selection. Hence no action should be taken. When ΔB_k is positive, the quality level is going to be increased, and when it is negative, the level is reduced. When $R(B) = q_{max}$ $\Delta B_k \leq 0$. Simply put, at the maximum buffer level an ABR scheme can only reduce or stay with the current quality level.

Figure 3.4 shows the model in both continuous and discrete form. As can be observed, in the discrete form the model is 'sticky', that is, the requested video rate remains unchanged for the duration of the buffer window ΔB_k. Furthermore, the higher the selected video rate, the stickier the model becomes. This has two advantages. First, the system can absorb short-term throughput oscillation, hence reduction in the fluctuation of video rate. Secondly, since the size of ΔB_k reduces with fall in video rate, the system can trade stability with rebuffering prevention, by making the system more responsive to the change in the buffer as the buffer occupancy reduces.

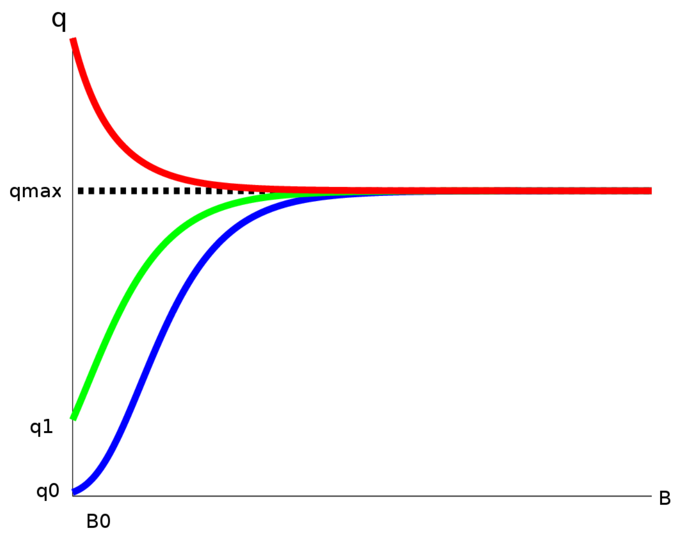


Figure 3.5: Plots of the rate map, each starting from a different value.

3.4 Behaviour of the Model

The rate map represented by Equation 3.11 is expected to be used by an algorithm designer to build an adaptation logic that guarantees the set of requirements earlier enumerated at the Section 3.2. This section presents an analytical analysis of the model, to show that it will indeed help a rate selection algorithm developer achieve his/her aim if used. Recall that a rate map is used to guide an ABR navigate its path from the lowest video rate to the highest video rate. To guarantee high average video rate and network utilisation the rate map must converge at the highest video rate, and remain there for the remaining duration of the streaming session. The next sections will discuss the video rate converges, stability, and the impact of α on them.

3.4.1 Convergence

We start by having a look at the pictorial representation of the rate map. Figure 3.5 presents three plots of Equation (3.11), with each plot starting from a different minimum video rate ($q_{min} = 100kbps, 2000kbps, 12000kbps$) but same maximum video rate (and $q_{max} = 8000kbps$). The most important observable characteristic of the curves is the nonlinear flow of the video rate as the buffer increases. Starting with a flatter slope, as the buffer occupancy increases the slope becomes steeper, halfway through the range of the video rates the slope began to flatten again. This pattern is same regardless of the starting video rate. Furthermore, we can observe that all the curves converge at the maximum video rate, though at different buffer positions.

To generalise this, the limit of $R()$ as buffer tends to infinity is taken, which results in $\lim_{B \to \infty} R() = q_{max}$. Put differently, q_{max} is asymptotically reached, independent of the initial value of the video rate (q_0). In summary, regardless of the starting video rate the maximum value that $R()$ will return, assuming an infinite buffer size, is q_{max}. To find the minimum buffer size that guarantees this convergence, we solve $R() = q_{max}$. It should be noted that any increase in the buffer size above the obtained value does not result in any rise in video rate. Therefore, barring any other consideration by an algorithm designer, this can be considered as B_{max}.

3.4.2 Stability

Having seen the rate map will converge at the right video rate, next, we will try to find out if the converged point is stable. You may recall that stability is part of the user requirement. The equilibrium of the model is when $\frac{dR()}{dB} = 0$. In other words when the system does not change its video rate. The result of equating Equation (3.6) to zero gives us two equilibrium points, $q^* = 0$ and $q^* = q_{max}$.

It is evident that when a client has not started requesting any video, it will stay in that state forever. However, it is interesting to investigate the behaviour of the model near $q^* = 0$. Since close to $q^* = 0$ the buffer level is low. When q is very small, αq^2 is small compared to $\alpha q q_{max}$. Therefore, equation (3.6) becomes $\frac{dR()}{dB} \approx \alpha q q_{max}$. We can infer from this equation that provided $\alpha > 0$ any small perturbation in the system state will result in an exponential growth of the video rate away from the current video rate, hence resulting in an equilibrium that is unstable.

The second equilibrium point is $q^* = q_{max}$. Again we are interested in what happens near this point. Let us assume that

$$\epsilon = q - q_{max}$$

. When we substitute $q = q_{max} + \epsilon$ into Equation 3.6, we get

$$\frac{dR()}{dB} = -\alpha\epsilon q_{max} - \epsilon^2 \tag{3.12}$$

However, if q is close to q_{max}, for all $\alpha > 0$ the ϵ^2 will be very small, therefore we have $\frac{dR()}{dB} \approx -\alpha\epsilon q_{max}$. Thus, small perturbation will decay exponentially, reverting to the q_{max}. Hence, the equilibrium $q^* = q_{max}$ is asymptotically stable.

3.4.3 Impact of the Evolution Constant on a Buffer

The constant α determines the speed of the video rate evolution. Figure 3.6 shows a plot derived from Equation 3.1 using different values of α. As can be seen, an increase in the value of α increases the amplitude of the path, which represents the maximum rate at which the video will evolve. In other words, the higher the value of α the faster the system converges at the maximum video rate. And since the quicker the convergence, the higher the average video rate, seemingly, an increase in the value of α is desirable. However, by changing the subject of the formula of Equation 3.10 to α we get the following equation:

$$\alpha = \ln\left[\frac{q(q_{max} - q_{min})}{q_{min}(q_{max} - q)}\right] . \frac{1}{q_{max}(B_t - B_{t_0}),} \tag{3.13}$$

From Equation 3.13 we can infer that the value of α only depends on the allowable buffer size, provided $q = q_{max} - \epsilon$, where the value of ϵ is very small compared to the q_{max}. This is because all other variables are constants. Therefore, the larger the buffer size, the smaller the value of α, hence the longer the convergence time. But since the larger the buffer size, the more stable the system is, there is a trade-off between the average video rate and stability. Equation 3.13 gives the expression that calculates the exact value of α. The derived value optimises both the video rate stability and its average value. Any value above the one computed in Equation 3.13 will reduce video rate stability and increase video rate and vice versa.

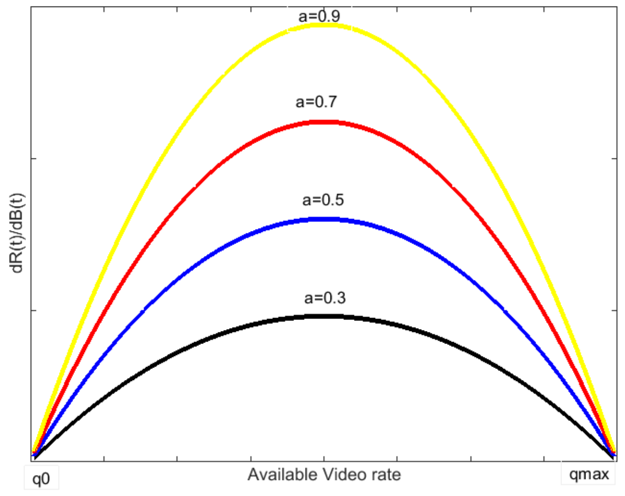


Figure 3.6: Derived trajectory of video quality evolution

3.5 Bio-inspired Interpretation of the Model

System analysis has been widely used, by ecologists, to help describe the behaviour of various organisms under different environmental conditions such as an increase or a decrease in food supply [153]. Typically, scientists start by isolating the 'the entities or parts which compose the system', then define the relationship between these components. Usually, the target is to model the mechanism that controls the dynamics of the ecosystem. Definitely, this requires a clear description of what a state is, and what constitutes a change it. For example, with the derived models, scientists can predict the impact of different events on the population of various species of animals.

Specifically, population dynamics is a branch of mathematical ecology that models the change in the population of different species as well as the processes that influence the changes, e.g. the availability of resources and the presence of the competing species [154].

To build these population growth models, some assumptions are typically made [155]. First, the size of the population, at any given time, represents the state of the system. Secondly, the state is affected by the exogenous variables, such as the availability of food or predators, and the previous state. For instance, the initial size of the population. Thirdly, the habitat can only support population up to its maximum carrying capacity (K). Fourthly, there is always a seed population that kicks start the process. The rate at which offspring are added is called the *birth rate* and the rate at which the animals die is referred to as the *death rate*. A plethora of these growth models exit in literature [156, 157, 158].

The rate at which population increases (r) is dependent of the birth rate (b) and the death rate (d). Therefore, $r = b - d$. Let assume the seed population is N_0, so that rate of change of the population is:

$$\frac{dN}{dt} = rN_0$$

.

However, the third assumption tells us that population growth is density dependent. In other words, the density of a population regulates its growth. This is basically because of the competition in the scarce resource. So the density dependence must reflect this fact. For example, ($\frac{K-N}{N}$), which measure the ration of the current population to the maximum carrying capacity of the habitat, or $(K - N)$ that measure the difference between the current capacity and the maximum carrying capacity, respectively, giving us either

$$\frac{dN}{dt} = rN_0\left[\frac{K-N}{N}\right] \tag{3.14}$$

or

$$\frac{dN}{dt} = rN_0[K-N] \tag{3.15}$$

The solutions to these differential equations give ecologists predictive models that help tackle problems, such as over fishing and saving animals that are on the verge extinction. But a look at these equations (Equation 3.14 and 3.15) shows a remarkable similarity with Equation 3.11. In fact, Equation 3.11 and 3.15 are exactly the same equation with different variables. Where N_0, K and r represent minimum, maximum and population evolution constant in Equation 3.15, q_{min}, q_{max}, α represent the minimum, maximum and video rate evolution constant This inspired us to reformulate the problem of video rate adaptation.

In this context, let us assume that the video rate is the species whose growth we are interested in. Furthermore, we suppose that the playback buffer is its habitat. Next, the rate at which the video rate of the incoming chunks changes is considered the birth rate, and the rate at which a player consumes content from the playback buffer is assumed to be the death rate. Recall from Equation (3.3) that the rate of content arrival is $\frac{c(t_i)}{q}$, which is considered to the birth rate. Furthermore, a player consumes content at a constant rate, precisely, one second of content is consumed every wall-clock second. Hence, a constant death rate. Assuming like in the natural habitat there are enough resources to sustain video rate up to the maximum video rate, that is $c(t_i) \geqslant q_{max}$. Therefore, the rate at which the buffer is filled (α) will be $\alpha = \frac{c(t_i)}{q} - 1$. However, like in anatural context, there is a seed video rate that reproduces at the rate of α to kick-start the growth. Hence,

$$\frac{dq}{db} = \alpha q_0.$$

Increase in video rate, just like the population is density dependent, the higher the video rate, the less we are inclined to increase, because throughput is not unlimited, and users are keen on an increase when video rate is relativity high. Therefore, a density dependence factor is needed, which will force the growth rate in the video rate to decrease as the maximum buffer level is approached. For example, $(\frac{q_{max}-q}{q})$, which measure the ration of the current video rate to the maximum available video rate, or $(q_{max} - q)$ that measures the difference between the current video rate and the maximum video rate can be used. These respectively give us either

$$\frac{dq}{db} = \alpha q_0 \left[\frac{q_{max} - q}{q}\right] \tag{3.16}$$

or

$$\frac{dq}{db} = \alpha q_0[q_{max} - q] \tag{3.17}$$

Clearly, Equation 3.11 and 3.17 are exactly alike. Just as biologists solve the differential equation (Equation 3.14 or 3.15) to get the predictive model used for various ecological studies, we can solve either Equation 3.16 or 3.17 to get a predictive model that can be used to predict the video rate to request, given any buffer size less than the maximum value. As in a natural habitat, the buffer size will determine a limit of the maximum video rate (q_{max}) a player can download. In this case, unlike in the wild, the maximum video rate is given (as defined in the MPD). Therefore, the task is mainly focused on finding the amount of buffer space required to guarantee the maximum video rate.

As seen in Equation 3.16 and 3.15, one of the advantages of the bio-inspired formulation is that some predictive models can be constructed with the same behaviour without much overhead. However, the one that exactly matches Equation (3.11) is the Verhulst-Pearl equation [156], perhaps the most well-known population growth model. Furthermore, if the streaming context changes, such that we are forced to modify the some of the assumptions made in deriving Equation 3.11, with the bio-inspired formulation there is no need to start modelling from the beginning all over again. For example, imagine we intend to develop an ABR that strictly serves users with a stable network and capacity the is a least twice the maximum video rate, and using the smart TV. In this scenario, we may what to speed up the convergence, with our current model that can only be done by using a large value of α, but recall this decreases the amount the buffer needed. And since buffer size is not an issue here, this may not be the most optimal solution. A better solution will be to change the point of inflexion of the curve, that is, the point which the rate of video rate increase begins to slow down. With bio-inspired formulation it is easy to try other growth curves, such as Gompertz model [158] with the point of inflexion at $\frac{q_{max}}{e}$.

3.6 Implementation

Most of our discussions so far are descriptive, including the derivation of the model. In other words, the derived model only provides us with some information on how things ought to be working. In this section, the proposed model is applied to two selected rate adaptation algorithms to demonstrate its applicability. First, the buffer-based algorithm proposed in [14, 23], and secondly the throughput-based ABR [22].

When modifying the implementation of the algorithm proposed in [14, 23], nothing is changed, except for the removal of the *reservoir*. Since as we may recall, the proposed rate map covers the entire period of streaming, that is, both the buffering and the steady state phases are taken care of by the model. Hence from the start, the algorithm relies on the proposed model. The summary of the algorithm is thus: the current video rate is increased to the next level only if the rate suggested by the proposed model exceeds the next higher available quality level. However, if the current video rate proposed by the model is below the next lower available video rate, the quality level is switched down. Otherwise, the algorithm retains its video rate (for a detail discussion of the algorithms see [14, 23]).

To retrofit the proposed model into the [22], the original algorithm had to be modified. It is worth noting that none of the changes affects the throughput related logic. To carefully map the original buffer dynamics, the playback buffer is divided into three phases. The first phase is when the video rate change is slow, with a threshold at B_{qt_1}. The next phase is when the video rate grows rapidly, which ends at B_{qt_2}. The third is when the video quality level increase reaches saturation, which starts at B_{max}. The threshold can be calculated thus:

$$B_{qt_x} = R^{-1}(q_{min} + \beta(q_{max} - q_{min}))$$

For $x = 1$ the $\beta = 0.1$ and for $x = 2$ the $\beta = 0.73$. The modified version of the throughput-based algorithm is presented in Algorithm 1.

Algorithm 1: Modified throughput-based ABR

```
input  : $c(t_k)_{k=1,\ldots t_e}$
         $B_k$ $\{B_k : 0 < B_k \leq B_{max}\}$
output: $q_{k+1}$: Next video rate
         $B_{delay}$
static runningFastStart := true;
$B_{delay=0}$
$q_{k+1} = q_k$
if  $runningFastStart$
  $\wedge\ q_k \neq q_{max}$
  $\wedge\ q_k \leq \alpha_1.c(t_k)$ then
    if $B(t) < B_{qt_1}$ then
        if $q_{k+1} \leq \alpha_2.c(t_k) \vee R(t) \geq q_{k+1}$ then
            $q_k := q_{k+1}$
        end
    else if $B(t) < B_{qt_2}$ then
        if $q_{k+1} \leq \alpha_3.c(t_k) \vee R(t) \geq q_{k+1}$ then
            $q_k := q_{k+1}$
        end
    else
        if $q_{k+1} \leq \alpha_4.c(t_k) \vee R(t) \geq q_{k+1}$ then
            $q_k := q_{k+1}$
        end
        if  $B(t) \geq B_{max}$ then
            $q_k := q_{max} \wedge delay := V$
        end
    end
else
    runningFastStart := false;
    if $B(t) < B_{qt_1}$ then
        if  $R(t) \leq q_{k-1}$ then
            $q_k := q_{k-1}$
        end
    else if $B(t) < B_{qt_2}$ then
        if $q_{k-1} \geq \hat{c}(t) \vee R(t) \leq q_{k-1}$ then
            $q_k := q_{k-1}$
        end
    else
        if $q_k \geq \alpha_5.c(t_k) \vee R(t) == q_{max}$ then
            $delay := V$
        else
            $q_k := q_k$
        end
    end
end
```

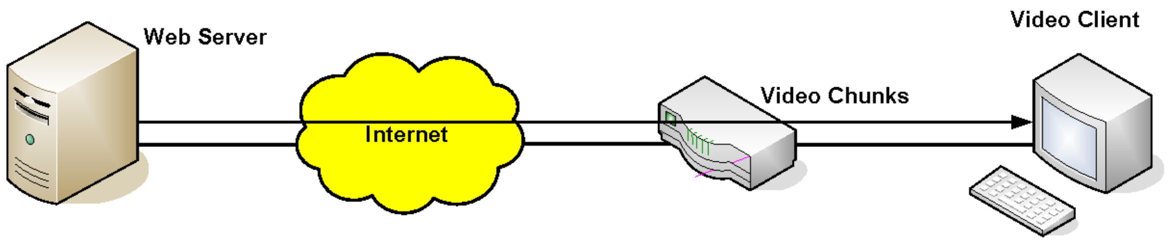


Figure 3.7: Experimental Set-up

3.7 Performance Evaluation

This section presents the experimental set-up and the performance evaluation metrics.

3.7.1 Experimental Set-up

The test-bed set-up is shown in Fig. 3.7. The client is connected to the Internet either via an Ethernet switch or using a 3G network. The web server is located at the Alpen-Adria-Universität Klagenfurt, which hosts the Big Buck Bunny dataset [159].

All the players used are implemented in Python, and run on top of Ubuntu 12.04.2 LTS. The host that runs the players also hosts Dummynet, tcpdump, lsof, and Wget. Throughout the wire-line experimentation, the maximum downstream available bandwidth was limited to $6mbps$. While for the wireless a "blue-sky" test was conducted. For all the buffer-based players, $B_{max} = 240s$, and for the player running the Huang et al. [14] original algorithm the reservoir was set to $40s$. For the growth constant of the proposed model $\alpha = 0.05$ is used throughout. While for both throughput-based algorithms (original and modified) the same configurations used in [22] is retained. Each experiment was conducted ten times, and the average result is presented. When more than one player is used or when a player and background traffic worked at the same time, all were run on the same machine.

3.7.2 Evaluation Metrics

To evaluate the impact of the proposed model on the two modified algorithms, the metrics tracking the user requirements presented at Section 3.2 are used. For this, several scenarios are used, with each designed to allow for a controlled measurement of one or more metrics. The metrics and their definitions are as follows:

- Rebuffering events: this is the total number of video freeze per streaming session [15].
- Average video rate: is the average of video rate played weighted by the duration each video chunk is played, calculated as $\frac{t_1 q_1 + t_2 q_2 ... t_n q_n}{t_n - t_1}$ and measured in kb/s [15].
- Instability: is the fraction of successive chunk requests, by a player, in which the requested video rate changes, measured at the steady-state [160].
- Utilisation of available network resource: is calculated by dividing of average video rate by the average network capacity [21].
- Convergence time: is the time taken for the video to settle at the sustainable video rate.
- Start-up Delay: is defined as the amount of time it takes a player to download a predefined number of chunks before the playback starts [147].

3.8 Result

This section discusses the result of the various test-bed experiments conducted in both wired and wireless environments. The purpose of these experiments is to evaluate how much improvement in QoE related metric, if any, is gained by the use of the proposed model.

3.8.1 Determination of Evolution Rate

In this experiment, the impact of evolution rate constant α on the performance of the model is investigated. Figure 3.8 shows the plot of Equation 3.13, with $q_0 = 50kb/s$ and $q_{max} = 8000kb/s$ The horizontal axis showing the buffer size and

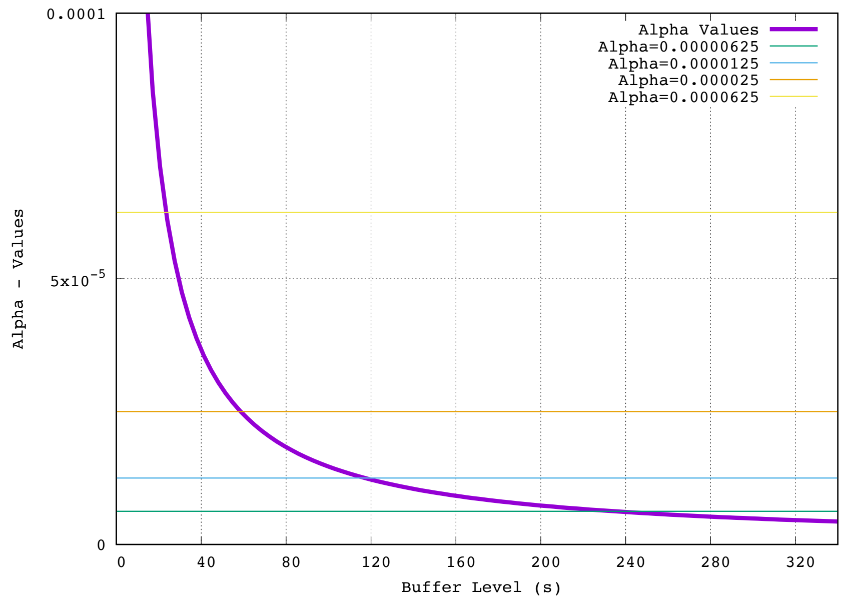
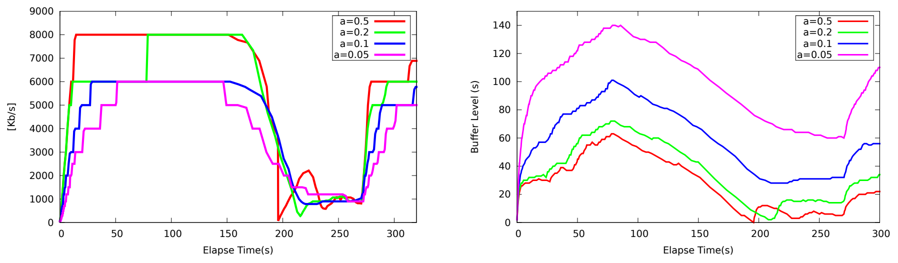


Figure 3.8: Experimental Set-up

the vertical axis showing the corresponding α values. As can be seen, for maximum buffer size of $240s$ the value of $\alpha = 6.25 \times 10^{-6}$ $(\alpha q_{max} = 0.05)$ indicated by the green line. To investigate the impact of different vales of α on the performance of the algorithms, we ran same experiment using different values of alpha, $\alpha = \{1.25 \times 10^{-6}, 2.5 \times 10^{-5}, 6.25 \times 10^{-5}\}$, with the modified buffer-based player. The buffer-based player is chosen because it allows the isolation the impact of model. For easy of representation, henceforth only the values of $\alpha q_{max} = \{0.1, 0.2, 0.5\}$ are used.

As shown in Fig. 3.9(a) for the values of $\alpha q_{max} = 0.5$ and $\alpha q_{max} = 0.2$ the system is very aggressive. The player downloads video rates that are more than what the available system capacity can safely handle. Additionally, when the bandwidth drops the player is not able to reduce the video rate to the sustainable level in time to avoid rebuffering (see Fig. 3.9(b)). The video stall is unnecessary since the capacity of the system could have sustained a lower video rate without any rebuffering. Furthermore, when the available capacity is low the player repeatedly undershoots its buffer, which not only increases the chances of rebuffering events but also increases the video rate



(a) Effect of different values of α on video rates. (b) Effect of different values of α on buffer evolution.

Figure 3.9: The impact of different evolution rate constant α.

Table 3.1: Effect of different values of evolution constant(α)

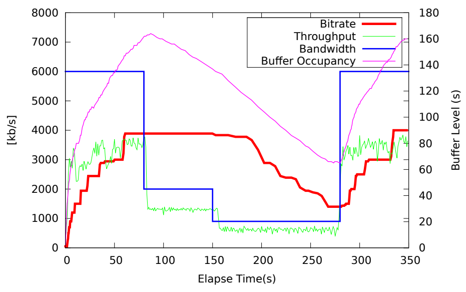
Evolution constant	Buffer-Undershoot	Convergence Time	$B_{max}(s)$ (%)
0.5	3	10	28
0.2	1	12	32
0.1	0	28	63
0.05	0	51	126

fluctuation. However, when the evolution rate constant is reduced to $\alpha q_{max} = 0.1$ and $\alpha q_{max} = 0.05$ the player is not only able to prevent video freeze but also the video rate can converge at maximum available bandwidth.

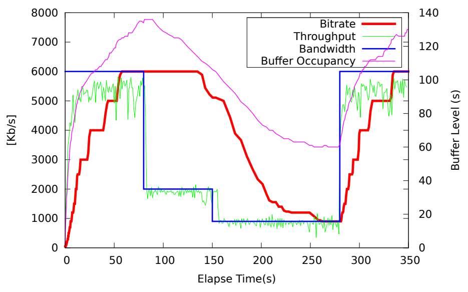
Table 3.1 summarises the impact of different values of evolution constant on the performance of the proposed model. It can be seen as the value of α increases both the buffer requirement and convergence time drop. However, this comes with an increased risk of rebuffering. This is in total agreement with our discussion in Section 3.4.3. In summary, it was found that when α is above the value calculated using Equation 3.13 the player is extremely aggressive, while the player is very stable when using the calculated value of α.

3.8.2 Bandwidth Sensitivity Analysis

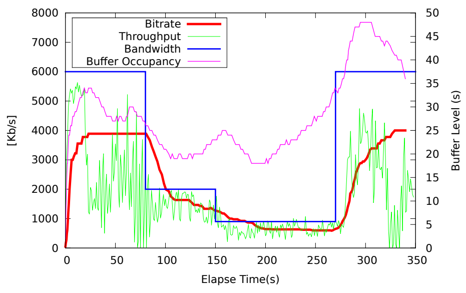
These experiments are aimed at demonstrating the elasticity of the proposed model, i.e. how it adapts to a rapidly changing bandwidth. As can be seen from Figure 3.10 (actual link capacity is plotted in blue), the streaming started with a maximum available bandwidth of $6mb/s$. At the $80s$ the bandwidth is dropped to $2mb/s$, then



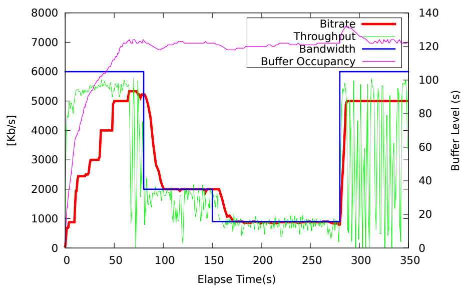
(a) Original buffer-based



(b) Modified buffer-based



(c) Original throughput-based



(d) Modified throughput-based

Figure 3.10: Video quality change, for both the original and the modified algorithms, operated in an environment with changing bandwidth.

at 150s it is dropped again to 900kb/s. Finally, at 270s it is raised back to 6mb/s and stayed there until the end.

The first thing to note is that the video rate of the segments downloaded by the player employing the proposed model (Figure 3.10(b) and 3.10(d)) converges at a higher video rate. In fact, the modified buffer-based player converges at exactly the system capacity (see Figure 3.10(b)). Table 3.2 shows that the modified players achieve a maximum video rate of 6mb/s and 5mb/s against the 4mb/s for the unmodified players. This translates to 100% and 85% throughput utilisation, which is an improvement of 33% and 18% utilisation compared to the original buffer-based and throughput-based players, respectively.

As can be observed, both throughput-based players suffer lower network utilisation than the corresponding buffer-based players. A look at Figure 3.10(c) and 3.10(d) shows that in both instances once the buffer level reaches the threshold value, both players activate the periodic download scheduling, a clear violation of the Policy 1, to stabilise the buffer. As can be seen in the green plot, this results in high throughput variation. Consequently, affecting the utilisation of both. Nonetheless, an improvement in the video rate of $854kb/s$ in the case of the buffer-based player and $433kb/s$ in the case of the throughput-based player is recorded.

3.8.3 Convergence Test

Next, a comparison of the convergence time between the modified players and the baseline players is presented. Two scenarios are investigated: upward convergence, that is, when there is an increase in capacity, and downward convergence, when there is a sudden drop in capacity.

Figure 3.11(a) shows when the bandwidth suddenly increases after being low for a considerable amount of time. As can be seen, after the $120s$, when the capacity suddenly rises, both players running the buffer-based algorithm converge at the right video rate, albeit at different times, validating our discussion at the Section 3.4.1. While it only took the player using the proposed model $65s$ to reach the convergence state, it took the original player three times longer (i.e. $165s$).

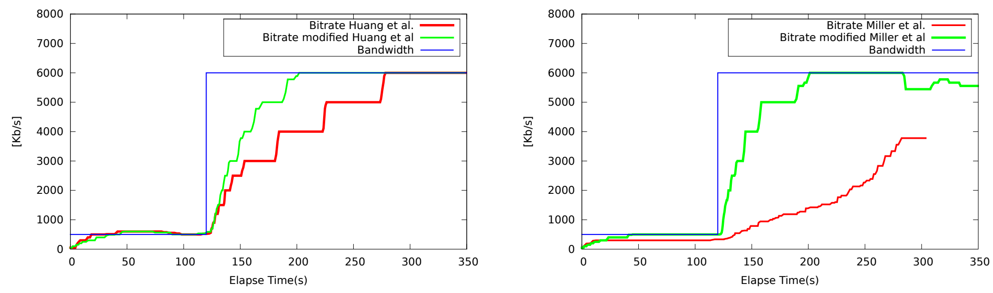
Furthermore, Figure 3.11(b) shows the case when throughput-based player are investigated. As can be seen, by using the proposed model, the convergence time is reduced by up to $80s$ in comparison to the original throughput-based player. It worth noting that while the modified throughput-based player converges at the actual system capacity, the original player is way behind by converging at $4mb/s$.

However, a player does not always converge to a high video rate. It can as well converge to a lower level as a result of a drop in the available throughput. Figure 3.12 presents such a scenario. When the bandwidth suddenly drops, it takes the player using the original buffer-based logic longer to converge, even though it is coming from a video rate that is a lot lower (see Figure 3.12(a)). That is $102s$ for the modified player against $146s$ for the original buffer-based algorithm, making the former more responsive to the change in throughput.

Furthermore, Figure 3.12(b) shows when the bandwidth suddenly drops, the player running the unmodified throughput-based algorithm was so aggressive in its reduction of the video rate that the player had to reach the lowest available video quality before

Table 3.2: Adaptation for variable bandwidth

Players	Maximum Video rate (kb/s)	Average Video rate(kb/s)	Throughput Utilisation (%)
Original buffer-based	4000	2982	67
Modified buffer-based	6000	3827	100
Original throughput-based	4000	2212	67
Modified throughput-based	5000	2645	85



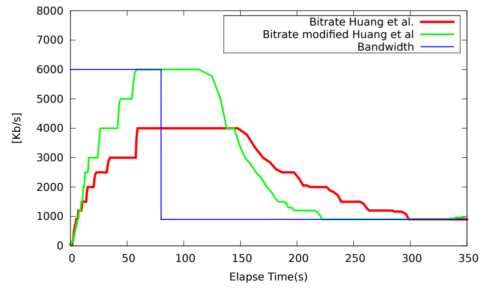
(a) Original vs modified buffer-based player (b) Original vs modified throughput-based player

Figure 3.11: Video quality convergence, for both the original and the modified algorithms, when bandwidth increases.

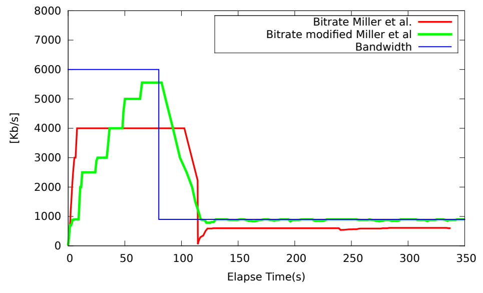
it later stabilises at a sub-optimal rate. Such a large amplitude in video quality change is detrimental to QoE. However, the player running the proposed model was much more conservative in its reduction and was able to converge at the appropriate quality level.

3.8.4 Start-up Delay

In all the experimentation conducted, the players are set to start playing after fifteen (15) chunks are downloaded, which translates to $B_t = 30$. Figure 3.13 shows the delay incurred by each of the players. As can be seen, both buffer-based players (modified and baseline) are able to start playing with a latency of less than $2s$, that is, about $1.7s$ after the first chunk is requested playback starts. Though they achieve similar performance, the original buffer-based player set its reservoir to $40s$ (the initial period when only the lowest video quality is requested), effectively downloading further twenty-three (23) chunks at the lowest video rate after the elapse of the start-



(a) Original vs modified buffer-based player



(b) Original vs modified throughput-based player

Figure 3.12: Video quality convergence, for both the original and the modified algorithms, when bandwidth.

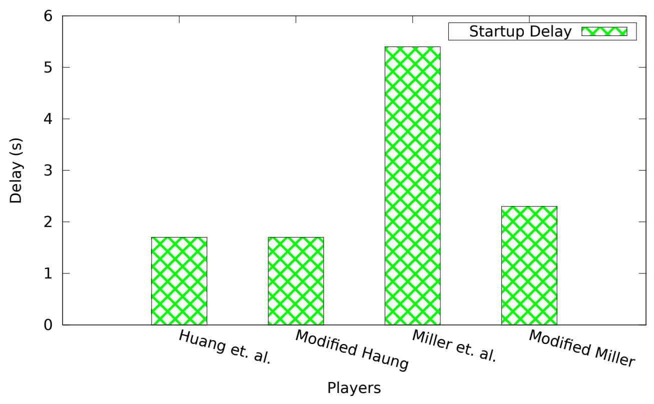


Figure 3.13: The start-up delay when $B_t = 30s$.

up period, when this needs not to be the case. However, the modified player starts a gradual increase in its video rate after the first nineteen (19) chunks are downloaded, i.e. it requests only four further chunks at the lowest video rate after the start of the playback. For the unmodified throughput-based player, the start-up delay is quite high, $5.4s$. But interestingly, the modified version reduces the start-up delay to about $2s$. While the former tries to strictly match video quality to the available bandwidth, hence downloading relatively high video rates at the start-up period the latter is more conservative, replenishing its buffer faster.

3.8.5 Stability

In the section, the stability of the proposed model is investigated. For this, experiments are conducted in a wireless environment. A MacBook Pro is used to run all

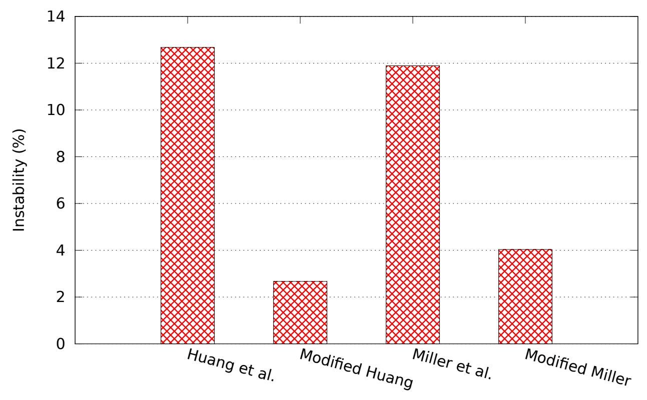


Figure 3.14: Video quality stability.

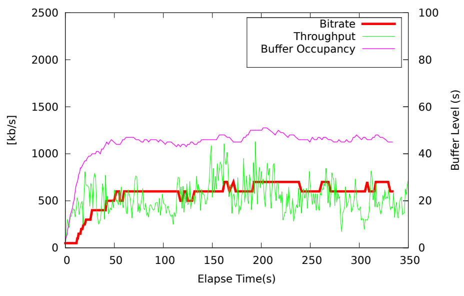
Table 3.3: Adaptation in Wireless Environment

Players	Maximum Video rate (kb/s)	Average Video rate(kb/s)
Original buffer-based	600	567
Modified buffer-based	1500	1247
Original throughput-based	700	536
Modified throughput-based	1500	1239

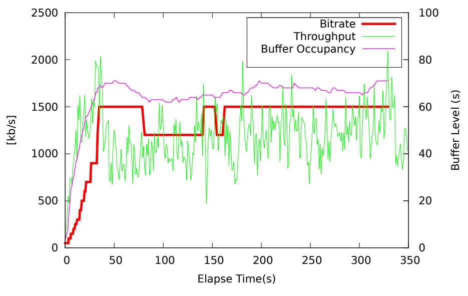
the players as and when required. The laptop is connected to the EE 3G network at Lancaster City Centre. The same server as in the case of the wired environment is used. All experiments were conducted within two days. The channel capacity, in all of the presented results, has not been restricted.

Video rate is said to fluctuate if the successive chunk requests by a player have different video rate. But this may not always be bad, at least while the system is ramping-up its video rate. Therefore, we only measure this when a player is at a steady-state. In this Chapter, we assume that a player has entered a steady-state, if it stays with the video rate, while ramping-up, for a period longer than the ΔB.

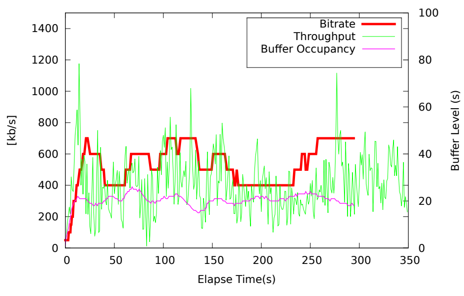
Figure 3.14 show that result of streaming with the players over a 3G network. As can be seen, both the original throughput-based and buffer-based players suffer a high degree of instability, at the steady-state the players are respectively 12.6% and 11.8% unstable. However, the instability is significantly reduced, when the proposed model is used, to 2.6% for the buffer-based player and 4.0% for the throughput player.



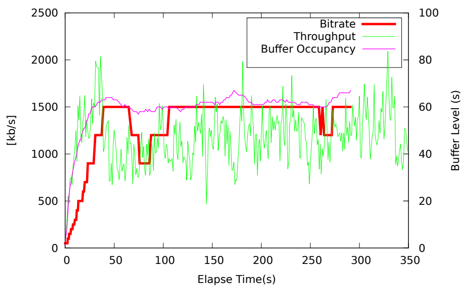
(a) Original buffer-based player



(b) Modified buffer-based player



(c) Original throughput-based player



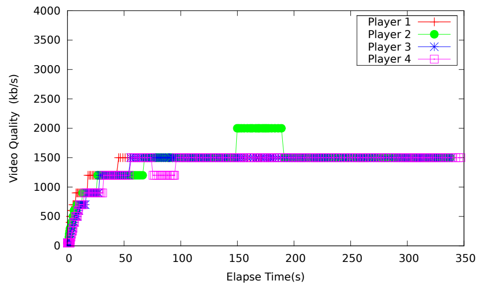
(d) Modified throughput-based player

Figure 3.15: Video quality change, for both the original and the modified algorithms, operated in a wireless environment.

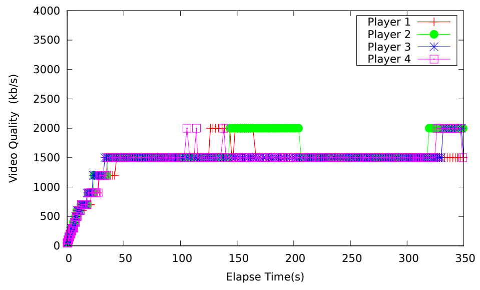
Furthermore, as can be observed from Figure 3.15(a) and 3.15(c) the maximum video rate attained by the original players are $600kb/s$ and $700kb/s$ respectively. However, the modified versions of the players are able to achieve $1500kb/s$ each (see Figure 3.15(b) and 3.15(d)). Importantly, this helps the modified players to achieve higher average video rate, with the modified players achieving average video rate that is at least twice of what the original players reached. The summary results are presented in Table 3.3. This result shows that the modified players can achieve a high average video rate while remaining very stable.

3.8.6 Fairness

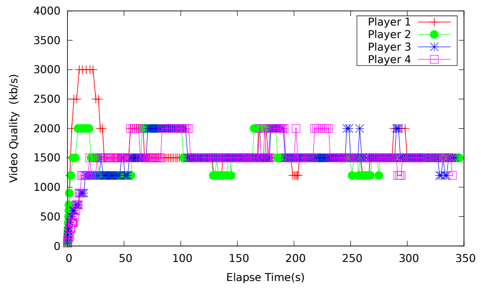
Another desirable property of streaming schemes is fairness towards other players and background traffic in the network. In this section, how fair the players running the proposed model is to other players and background long-live TCP traffic is investigated.



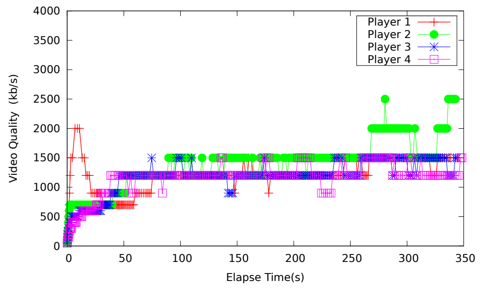
(a) Original buffer-based player



(b) Modified buffer-based player



(c) Original throughput-based player



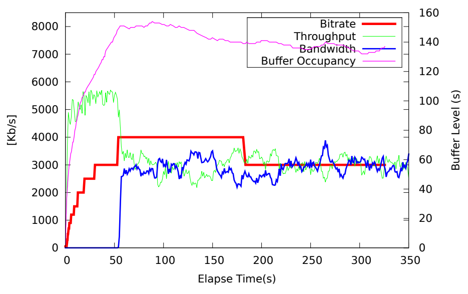
(d) Modified throughput-based player

Figure 3.16: Four players streaming at the same time.

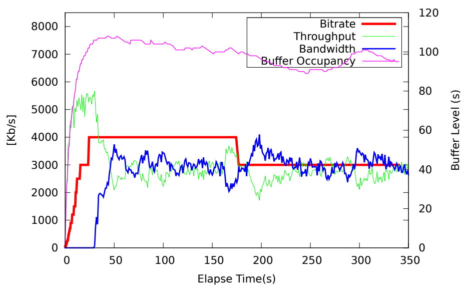
3.8.6.1 Multiple Players

Each time during the experimentation, four players all using the same implementation of the proposed model or the original versions are run at the same time. In each case, the maximum bandwidth of the bottleneck link is set to $6mb/s$. In the case where the players are fair to one another, we expect that they should equally share the available bandwidth since all the players are connected to the same network and are also running on a similar device.

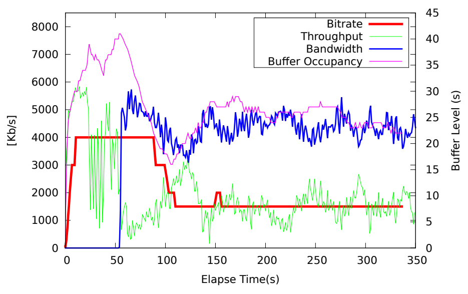
As can be seen from Figure 3.16(a) and 3.16(b) as players compete, none of the buffer-based players used more/less than $1.5mb/s$, which is the fair share. Furthermore, as can be observed in the figure, the players achieve this with a high level of stability. However, the case of the two throughput-based players is slightly different, both players were able to reach the fair bandwidth, but it comes at the cost of an increase in instability (see Fig. 3.16(c) and 3.16(d)).



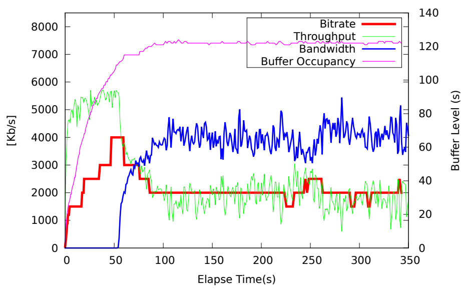
(a) Original buffer-based player



(b) Modified buffer-based player



(c) Original throughput-based player



(d) Modified throughput-based player

Figure 3.17: Video quality change as a player compete with background TCP traffic.

3.8.6.2 A Player and Background Traffic

In this section, the impact of background TCP traffic on all the players is investigated. To do this, the case where a player and background traffic (file download from the same server) compete is investigated. The download starts $30s$ after the start of streaming.

As can be seen from Figure 3.17 in all the investigated scenarios the players can use, almost, the entire available bandwidth (see the achieved throughput in green). However, as soon as the background traffic is started, the achieved throughput starts to drop until an equilibrium is reached gradually. The buffer-based players (both the baseline and modified versions) fairly share the available bandwidth, that is, each uses about $3000kbps$. Furthermore, it is worth noting that the drop in video rate does not affect the stability of the players. In other words, both buffer-based players are able to avoid video rate oscillation in their download even though the TCP throughput fluctuates (see Figure 3.17(a) and 3.17(b)). However, the original throughput-based player requests video chunks lower than its fair share ($1500kbps$) under-utilising its fair share of the available bandwidth by about 50%. The modified version improves the video rate to $2000kbps$, however, still not an optimal utilisation of the available bandwidth.

3.9 Summary

Designing an effective adaptive bitrate selection algorithm requires a careful selection of the restricted set of states that, at any given time, the system can be in. Equally important is the nature of the patterns of the state transition when the context changes. The chapter started by reviewing the classical framework of the interaction between the ABR components previously presented in the previous chapter, which makes a distinction between the internal and the external part of the ABR. With the internal part being composed of throughput estimation, buffer management, scheduling, and adaptation modules. The external part is made-up of user experience module and user context.

The classic framework is then reformulated to get a new framework, which breaks the cyclic relationship between throughput estimation, scheduling, and buffer management subsystems. Based on this, the definition of a state in ABR is revisited. Since states are not considered in isolation, a model of the state evolution trajectory is developed from the patterns of video rate transitions that are known to affect user experience. The behaviour of the model, such as convergence and stability were then analytically analysed. Using the bio-inspired method, the same problem is formulated. The benefits of using this design methodology are then discussed.

Since the model presented is descriptive, to demonstrate how it can be used in practical systems two algorithms: a buffer-based and a throughput-based are modified to work with the proposed model. To evaluate the performance of the proposed model, experiments were conducted over the Internet, using both wired and wireless connections. Results of the evaluation show that the users streaming with the modified players experienced an increase in the average video rate, capacity utilisation, and stability. While at the same time a reduction in both start-up delay and the convergence time were achieved. This happens without any negative impact on the player's fairness both to other players and background traffic. Interestingly, not even a single instance of video freeze is observed.

However, there are still some issues with the proposed scheme that require improvement. First, using throughput estimate as a constraint forces the proposed model to use larger buffer size, which may be unavailable in some resource-constrained devices. Secondly, any video rate selection decision that does not take the evolving state of the user experience into consideration is bound to result in a sub-optimal decision.

The next chapter will propose a solution to these issues. First, a new framework will be presented that shall make the user experience an active component of an ABR, such that any decision taken by the adaptation logic will now take into consideration the prevailing QoE state in addition to any other system parameter that it may take. And then makes the throughput estimate an adjustment factor. Based on the reformulation of the problem, a new set of models that defines the relationship and interactions between the various components of ABR is then presented.

Chapter 4

Dynamics of Video Quality

4.1 Introduction

In the previous chapter, some assumptions that can compromise QoE have been made. By restricting the problem space to the system level dynamics, the adaptation module decides on the video rate to request, from a server, without feedback from the user experience module. However, without taking into consideration the evolving state of QoE, it will be difficult to build an algorithm that truly enhances the user experience. Secondly, in an environment, such as wireless context, where the network capacity is both low and fluctuating continuously, relying on buffer only forces the system to choose between large buffer size or an increase in video rate oscillation. Because most of the streaming devices used in the wireless mobile environment are resource-constrained, the previous model might help in improving the performance of ABR algorithm with regards to some QoE metrics; it will be of little help in reducing the video rate oscillation.

This chapter revisits how best to model the relationship and the interactions between the various components of an ABR. It starts by reformulating the general behaviour of ABR schemes, but this time the user experience module is included. Then a unified framework that takes into consideration state of all the components of the ABR when making video quality selection decision is developed. Based on the specification of the proposed framework, a set of models describing the system

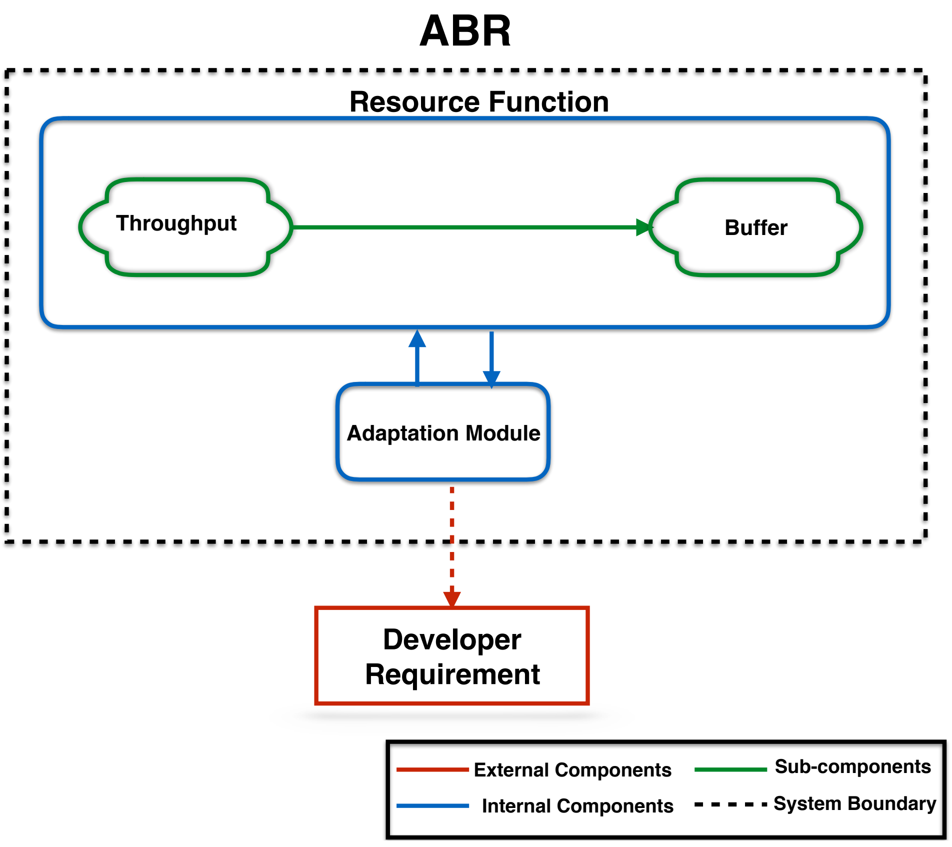


Figure 4.1: The updated activity diagram of the framework 1

when operating within either a resource-abundant context or resource-constrained environment is presented. After this, the behaviour of the model, such as stability, convergence, and rebuffering are analysed.

4.2 General System Analysis

In the previous chapter, the scheduling function was replaced by a policy which simplified our design. Figure 4.1 presents the Framework 1 without the scheduling function. As can be seen, only the dynamics of the two remaining components affect the decision made by the adaptation logic. Next, we discuss the general behaviour each of the two remaining components interacting with the adaptation module.

4.2.1 General Dynamics of Video Chunk Request

A typical system state is determined by two classes of the variable: the controlled and exogenous variable. While a typical system designer has control over the former, this is not the case concerning the latter. Based on this fact, the general dynamics of video rate switch, in HAS, can be succinctly represented by the following system state equation:

$$\dot{x} = u(t) \pm z(t) \tag{4.1}$$

where $u(t)$ is the input to the adaptation controller, which is used as a situational indicator. For example: buffer occupancy as seen Chapter 3 and [14]; throughput in [19, 18]; and power level in [70]. The variable $z(t)$ is the exogenous component of the ABR, that is, the part of the ABR service that a typical content provider has no control over. For instance, in the previous chapter, it is assumed that a content provider does not influence any factor that affects the user experience except the video rate. While $\dot{x}$ represents the evolution of the system, e.g. the rate at which buffer change with respect to time [24] or the rate of video bitrate change with respect to buffer occupancy.

4.2.2 General Dynamics of User Experience

The primary objective of a typical ABR scheme designer is to ensure a high level of QoE, which is defined as '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' [139]. The following definition can be modelled thus:

$$\dot{UX} = DI(t) - \lambda AI(t). \tag{4.2}$$

Where DI is the delight index that tracks the degree of user satisfaction with the system. AI is the annoyance index at any given time t after the streaming has started, which represents the cumulative annoyance a user suffers while streaming. And λ is a constant that captures the level of user's tolerance to a degradation in the video quality. This depends on 'user's personality and current state'[1].

The minimum level of satisfaction that guarantees a user does not abandon a streaming session can easily be found by equating Equation 4.2 to zero. The solution tells us that provided $DI(t) \geqslant \lambda AI(t)$ a user will continue streaming and may only abandon the session for any other reason but poor quality. Since the video quality perceived by a user does not always equate to the video rate, in this chapter and beyond, a distinction would be made between the *video quality*, which is used to refer to the user perceived video fidelity and the video rate, which is the bitrate of the requested chunk. Many factors are known to have an impact on the QoE, in this chapter we restrict ourselves to the following:

1. average video quality [161],
2. video quality fluctuation [142, 162],
3. number of rebuffering events [15].

Apparently, from the above list, only the increase in video quality level improves user delight. Therefore, we define the delight experienced by a user at any given time during a video streaming session to be a function of the video quality, as thus:

Definition 4.1

$$DI(t) = f(video\,quality)$$

However, an increase in any of the remaining three (3) QoE metrics negatively affects the QoE. Primarily, an ABR architect always aims at the absence of rebuffering even though it is not an achievable target. This is because the available network

[1]The modelling of this constant (λ) requires an advance psychological study that is out of the scope of this thesis, in future, we intend to further work on it.

capacity can go below what even can sustain the minimum available video rate. Therefore, a more realistic aim should be to avoid any unnecessary video freeze. A rebuffering event is called ‘unnecessary’ when it occurs while the network capacity is greater than at least the minimum video rate.

Policy 3

$$Rebuffing\,Event = 0$$

$$\textit{subject to} \quad throughput > minimum\,video\,rate$$

If we build a model that guarantees Policy 3, we can be sure that for any rebuffering event that occurs, the annoyance suffered, is unpreventable. Therefore, the annoyance a streaming user suffers that is directly attributable to the system design will come from the video quality fluctuation. Hence, $AI(t)$ can be defined as thus:

Definition 4.2

$$AI(t) = f(video\,quality\,fluctuation)$$

Simply put, provided Policy 3 is achieved, the system designer is only concerned with video quality fluctuation.

$$y(t) = R(u(t)) \tag{4.3}$$

With the existing architecture the adaptation function $R()$, which is the video rate selection function that maps input, typically takes the controlled variable as the input of the Equation 4.1, to the output $y(t)$. The output is generally the video rate of the chunk $i + 1 \leqslant l$ (see the Definition 3.2). Various methods have been used to realised this function e.g., heuristics [22], control theory [24], and machine learning [122].

4.3 Unified Framework

The decision to request a particular video rate invokes several side effects. For example, increase in video quality, change in buffer level, increase in the video quality fluctuation etc. Some of these events may result in a short-term improvement and perhaps a long-term negative impact on QoE, and vice-versa. With the Classic Framework (Figure 3.1) or Framework 1 (Figure 3.2) it is difficult for an adaptation logic optimises QoE since it lacks a complete picture. Typically, the predictive model derived from these frameworks is represented as thus:

$$R() = f(u(t)). \tag{4.4}$$

Where $u(t)$ is the controlled variable of the Equation 4.1 and the output is generally the video rate of the chunk $i + 1 \leqslant l$ (see the Definition 3.2). This section proposes a new framework called *Framework 2*. The framework is aimed at an ABR algorithm designer with the information needed to make an optimal decision. In other words, the framework shall allow for the construction of a function $R()$ that takes both two presented state equations (Equation (4.1) and (4.2)) into consideration.

On comparing Equation (4.1) and (4.2) a salient fact emerges. By mapping the right-hand side of two equations, we can see a direct relationship between the change in state at the system level ($\dot{x}$), and at the user experience plane ($\dot{UX}$). This relationship is what makes it difficult to design an ABR algorithm that optimises QoE without directly tracking the changes in user experience.

Considering the left-hand side of the two equations, there is a mapping between the system's controlled input ($u(t)$) and the user delight (DI). This mapping mandates the establishment of a relationship between the system's input and the video quality perceived by a user. This is consistent with the Definition 4.1, which relies on the fact that increase in video quality delights users. To guarantee this, a user must have control over the variable that affects the $DI(t)$ most. But the challenge here is that there are a plethora of video quality metrics, to avoid tightly coupling the framework to anyone of them video rate is used as a proxy for the video quality. However, this requires a function expressive enough to map the video rate to any of available objective video quality metrics.

Definition 3.3 gives a restricted definition of the system state. Therefore, in this

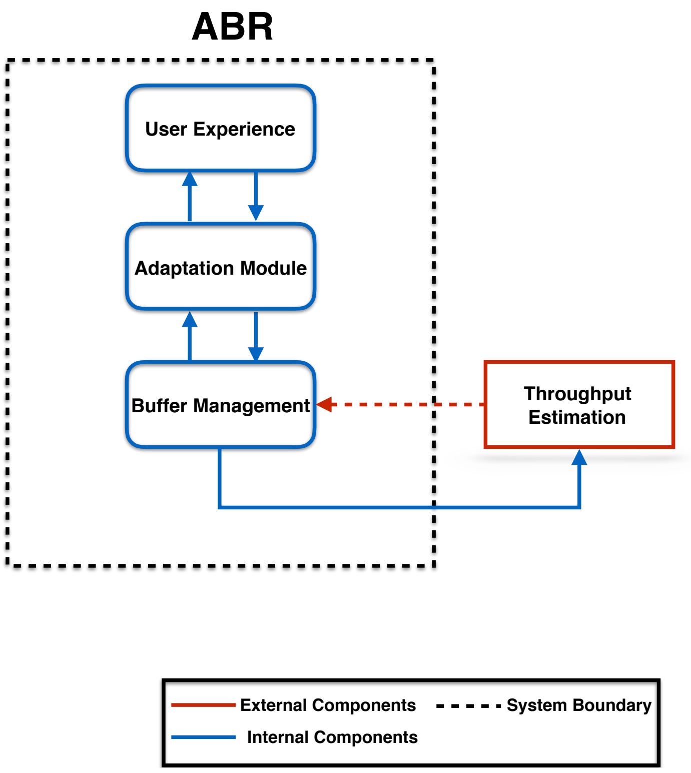


Figure 4.2: The activity diagram of the proposed unified framework

chapter, we use the broader Definition 3.1. Since as seen in the previous chapter, the buffer level at any time during the streaming session can be controlled and accurately measured it should be the controlled input to the system. Therefore, a model that formalises the relationship between the buffer state changes and video rate taking into consideration both the current video rate and the maximum rate, and how the change in video affects user experience is needed.

Finally, comes the mapping between user annoyance (AI) and the exogenous ($z(t)$) component of the system state equations. This mapping shows that the exogenous component (which is the part outside of the control of an ABR designer) is responsible for most of the annoyance a user suffers. As can be observed from Definition 3.3, the only other state variable is the throughput. However, in Section 2.3 it has been shown that the dynamics of TCP throughput is the leading cause of video quality fluctuation, which in turn is seen in Definition 4.2 to be the primary source of user annoyance. Furthermore, a typical content provider has little or no influence over the TCP throughput of the last-mile channel. Hence TCP throughput, not user experience should be the exogenous component. Instead of directly trying to control it, Policy 1 presents a policy that mitigates its negative impact, but when a streaming client operates in a wireless environment where throughput always changes this is not the optimal solution. Therefore, what is required is to model the extent to which fluctuation in throughput affects the likelihood of the system reaching its target.

Figure 4.2 presents the outcome of the preceding discussion. It shows the activity diagram of the Framework 2. The first thing to note is that compared to the previous frameworks especially the Classic Framework, the system is now greatly simplified. As can be seen, TCP throughput is now an external component. Hence no attempt is made directly at controlling it. By adhering to Policy 1, TCP is allowed to function as designed. Note, this does not do away with the need of accurate throughput estimation. In fact, it helps in such an estimate, since it allows TCP to converge at its actual capacity without being dragged down by the unnecessary feedback loops.

The buffer management module is now the entry point. However, there is a dashed line from throughput estimation module to buffer management indicating that though the throughput is not an internal component, it is an exogenous variable used for adjusting the system state. The most important change to the previous frameworks is that in Framework 2, the user experience function is an active internal component providing input together with the buffer management to the adaptation logic. This allows any algorithm built to adapt video quality in a manner that jointly optimises both resource utilisation and user experience metrics. It should also be noted that the framework can easily be modified to consider other factors. For instance, an ABR that may desire to optimise power usage can easily include battery level as an adjustment factor in addition to the throughput.

4.4 System Modelling: Over-provisioned Network

This section presents the model of relationships and the interactions between the various components of the Framework 2 when the system is operating in an environment with abundant resources. Then followed the analysis of the behaviour of the model.

First, suppose that a client can pick any bitrate between the q_{min} to q_{max}. This is an oversimplification of the reality since HAS constrains clients to choose video rate only from a discrete set. However, treating the video rates as continuous variables greatly simplifies the modelling process.

As discussed in Section 3.3.1, HAS clients rely on an estimated throughput for the purpose of rate selection. Let $\hat{c}_{i-1}$ denote the estimated per chunk throughput of chunk $i-1$ at time t with $C(t)$ being the actual system capacity. Now, suppose $c(t_i)$ is the average of h chunks, at time t_i, calculated thus:

$$c(t_i) = \frac{1}{t_i^s - t_{i+h}^e} \int_i^h \hat{c}_i dx \tag{4.5}$$

Let us assume that the available TCP throughput is at least twice the maximum bitrate, i.e. $c(t_i) \geqslant 2q_{max}$ as required by Wang et al. [52] for an artifact-free TCP-based streaming service[2]. In summary, the impact of the exogenous [3] component on the system is negligible. Therefore, in this scenario $z(t) \approx 0$ (see equation (4.1)). Though, in theory, it is possible to start and remain at the q_{max}, as stated in Section 3.2 this will result in an unacceptably long start-up latency. Hence, what may be required of a good HAS player is to converge at q_{max} at the shortest possible time and remains there for rest of the streaming session.

4.4.1 Dynamics of Video Rate I

The video rate evolution trajectory proposed at the Section 3.3.1 is symmetrical concave path with amplitude at $q_{max}/2$. This makes the model behave similarly regardless of whether the buffer level is high or low. However, when buffer occupancy is low, we want an ABR algorithm to be more responsive to the change in the network condition since the chances of buffer depletion are higher. And when the buffer level is high, an ABR designer is more likely to be interested in preventing video rate oscillation, because at this point the probability of buffer depletion is low. Therefore,

[2]At the Section 4.5 we shall relax the two assumptions.

[3]Exogenous, are the components that are out of the control of a typical ABR designer.

the rate at which video rate is adjusted should be more modest.

In order to ensure that these observations are taken care of, the video evolution trajectory is now improved. Without loss of generality, let us assume that a client starts streaming with the minimum video rate. After the receipt of chunk $i > 1$, in compliance with the Policy 1, the client requests chunk $i + 1$ with video rate $q_{k+1} = \gamma q_k$, where $\{\gamma : \ 0 < \gamma < 1\}$, expect when the buffer is full. Just like the previous case, $\dot{x}$ will not be dependent on time anymore but on the buffer occupancy.

$$\frac{dq}{dB} = \gamma q. \tag{4.6}$$

However, since the relationship between the video quality perceived by a user and the video rate is nonlinear, a constant rate of video bitrate change is not the most optimal choice. Therefore, to tightly fit the manner at which the video rate is changed to the pattern of video quality perception of the user, a mapping between the two metrics is needed. To do this, let suppose that $\Theta(q)$ is a function that takes a video rate and return an equivalent video quality in any of the variety of the objective quality metrics, such as Mean Opinion Score (MOS), Structural Similarity Index (SSIM) etc.

$$\Theta(q) = \{\theta_1, \theta_2, \theta_3 ... \theta_n\}$$

.

Usually, $\Theta(q)$ is derived using a regression analysis (see Section 2.8 for detail). Figure 4.3 shows $\Theta(q)$ that maps the video bitrate to the corresponding quality level using SSIM for different video resolutions, together with the derived generalised equation as presented in [1], which corresponds to the following equation:

$$\Theta(q) = aq^b + c. \tag{4.7}$$

where $\{\Theta(q) = \theta : \ 0 \leq \theta \leq 1\}$, when $\theta = 1$ we have maximum possible quality. And a,b,c are video resolution dependent constants. As can be observed from the Figure 4.3 as the video rate increases a point is reached when the video quality experienced

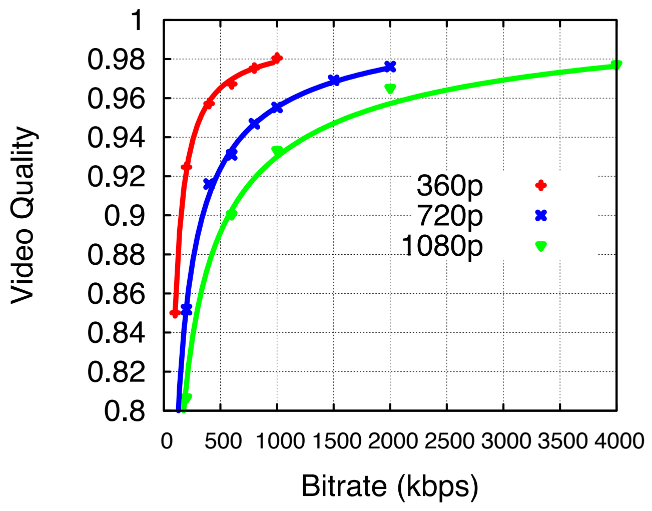


Figure 4.3: Mapping of bitrate to video quality [1]

by a user saturates. A corollary of this fact, is that, it requires more bits to raise the video quality perceived, by a unit, when the requested video rate is already relatively high than when it is low. This fact may not be of much concern if not because Cranley et al. [28] have shown that when video quality is relatively high viewers do not appreciate a further increase the video quality. In summary, *it costs more to increase video quality at the point when the increase is least needed.* Therefore, it will be more appropriate to have an adaptive video rate evolution constant γ, such that player will be more conservative with its video rate switch as the video rate increases, and more responsive when it is low. Hence, it is now suggested that the value of γ should reduce with a value η as buffer increase, thus the rate at which γ changes with respect to buffer changes is:

$$\frac{d\gamma}{dB} = -\eta\gamma. \tag{4.8}$$

Solving Equation (4.6) and(4.8) we have;

$$\gamma = \xi - \frac{\eta}{q} \ln \left[\frac{q_{max}}{q}\right] \tag{4.9}$$

where ξ is constant of integrating Equation (4.8). Thus, we can infer from Equation (4.9) that the value of γ only depends on the current video bitrate q since all other

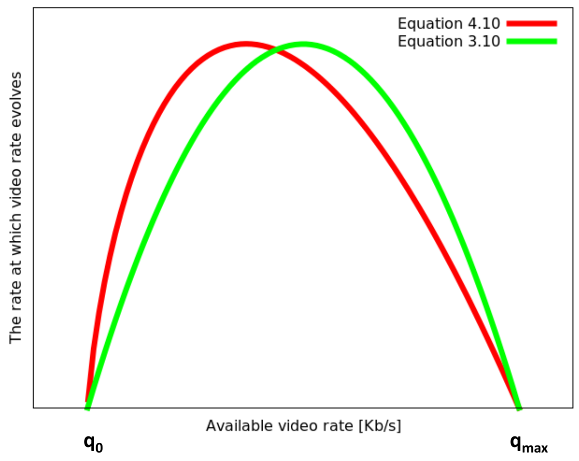


Figure 4.4: The Plot of the two trajectories of video rate evolution

variables are constant. And the relationship is inversely proportional. Simply put, as the video rate increase the value of theγ decreases. Using this information, if we assume $\alpha = \xi - \frac{\eta}{q}$, we can substitute the Equation (4.9) into (4.6), and the final expression of the video rate evolution trajectory, as thus:

$$\frac{dq}{dB} = \alpha q \ln \left[\frac{q_{max}}{q}\right] \tag{4.10}$$

In Figure 4.4, the plots of the two video rate trajectories are presented. The green plot is the one presented in the Equation (3.1) and the red plot is the improved version represented by the Equation (4.10). As can be observed, the plot of Equation (4.10) is not symmetrical but skewed to the left. Hence, the new turning point is no more $\frac{q_{max}}{2}$. To find the point at which the system is fastest, Equation (4.10) is equated to zero and the solve, which gives us $\frac{q_{max}}{\mathrm{e}}$. Furthermore, as can be seen, the new plot has a steeper slope when $q \leqslant \frac{q_{max}}{\mathrm{e}}$. This means before the turning point the system is faster than afterwards.

So far, we have seen when the system is expected to be fast and when it is expected to be slow. Given this information, just like in Section 3.3.2, we now derive a new rate map, henceforth called *Rate Map 2*. To do this, all that is needed is to solve the differential equation (4.10), with $B_t = 0$, when $q = q_{min}$. Please take note henceforth $y(B)$ will be used to represent $y(t)$ because as seen earlier the system depends on

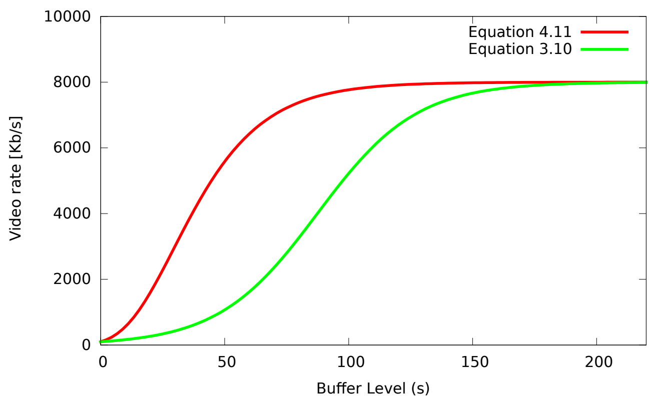


Figure 4.5: The Plot of the two trajectories of video rate evolution

buffer changes, and the buffer is calibrated in units of time. The solution gives us the following equation:

$$y(B) = q_{max} \mathrm{e}^{\left[\ln\left(\frac{q}{q_{max}}\right)\mathrm{e}^{\alpha \Delta B}\right]}. \tag{4.11}$$

Figure 4.5 shows the rate map derived from Equation (4.11) in red plot and the one derived from Equation (3.11) in green plot. There are several things to note, in these two plots. The first thing to observe is that both have an s-shape. This tells us that the pattern of video rate transition will be the same, except that we can see the real plot has the steeper slope at the beginning, which makes it faster. Consequently, Equation (4.11) converges faster and requires less buffer space.

4.4.1.1 Convergence

Since throughout the streaming session $c(t_i) > q_{max}$ applies, by transitivity it can be implied that after every chuck download $c(t_i) > q_k$ is true. Hence, the buffer level will continue to rise regardless of the requested q_k provided that $B_t < B_{max}$. This allows the client to progressively increases its video rate using $y(B)$ presented in Equation (4.10), such that given any buffer level the rate map returns an appropriate video rate to be requested. However, since the rate map cannot return a video that higher than the maximum video rate, on reaching the q_{max}, it stays there. This can be confirmed analytically by finding the limit of the Equation (4.10), given an infinite buffer size, which results in:

$$\lim_{B \to \infty} y(B) = q_{max}$$

4.4.1.2 Stability

It is easy to show from Equation (4.10) that the model has two equilibrium points, the first at $q^* = 0$ and the second at $q^* = q_{max}$ with the latter point corresponding to the convergence rate. Since it is most likely that the system is going to be disturbed, then we will be interested in finding out how the system behaves in case of such perturbations at the neighbourhood of these two equilibrium points. Near the $q^* = 0$, that is at $q_k = q^* + \epsilon$. Clearly, $q_{max}/q_k > 1$ which makes $\ln(\frac{q_{max}}{q_k})$ from Equation (4.10) positive, thus $\dot{x}$ increases. Therefore, $q^* = 0$ is an unstable equilibrium point. For the second equilibrium point, since there is no $q_k > q_{max}$, then can only investigate the case when $\{q_k : 0 < q_k < q^*\}$. Hence, for all values of q_k we have $\ln(q_{max}/q_k)$ being positive, implying that $\dot{x}$ is increasing, making the second equilibrium point globally and asymptotically stable.

4.4.2 Dynamics of User Experience I

Anytime video rate is changed, the quality of the video is affected. Fortunately, when the network capacity is more than the maximum video rate, this change in video rate is only isolated to the ramping-up period. Because as seen, the Rate Map 2 is guaranteed to converge at the maximum video rate and remains there for the rest of duration of the streaming session. By implication, at the steady state, the highest video quality θ is assured, which can be calculated using Equation 4.7. Furthermore, as we have already seen, the influence of the exogenous component is negligible. Therefore, the annoyance a user may likely suffer while streaming video, that is $\lambda AI(t)$, which as seen at the Definition 4.2 depends on only video rate fluctuation, will be negligible compared to value $DI(t)$.

Therefore, the change in user experience will be dominated by the DI, such that

$\dot{UX} \approx DI(t)$. From Definition 4.1, the delight experienced by a user streaming is only dependent on the video quality of the chunks being streamed. It turns out that all that is needed to measure the video quality of streaming a series of video chunks is to compute the running average of the per chunk video quality of the requested chunks, over the time interval of interest. However, Karapanos et al. [163] have shown that user's memories fade with the passing of time. To account for this fact, an exponentially moving average is used instead.

$$HI = h\theta_{k-1} + (1-h)\theta_k \tag{4.12}$$

where h is the coefficient that represents the weighting of the degree memory decrease.

4.5 System Modelling: Under-provisioned Network

In reality, it is not always possible to meet the criteria set by Wang et al. [52] on which the idealised case based its assumption on, in the previous section. In many environments, for example, 3G, the available bandwidth may not be sufficient to sustain the maximum video bitrate that a typical content provider may want to offer. This can result in a drop in the perceived quality of a streaming session. Furthermore, as can be seen in Figure 4.6, the throughput of a typical last-mile channel fluctuates significantly from the average value. Implying that we cannot afford to disregard the external factors.

4.5.1 Dynamics of Video Rate II

As a consequence of the preceding discussion, the earlier assumption of the abundance of the network resource is now relaxed. Let us suppose that $c(t) \leq q_{max}$. In this situation, the best that can hope for, is for a HAS player to converge at $q_s \leqslant c(t)$, where $\{q_s : \ q_{min} \leqslant q_s \leqslant q_{max}\}$. Since as you may recall from Policy 2, an ABR can not request video rate higher than the measured throughput. Let us call q_s the

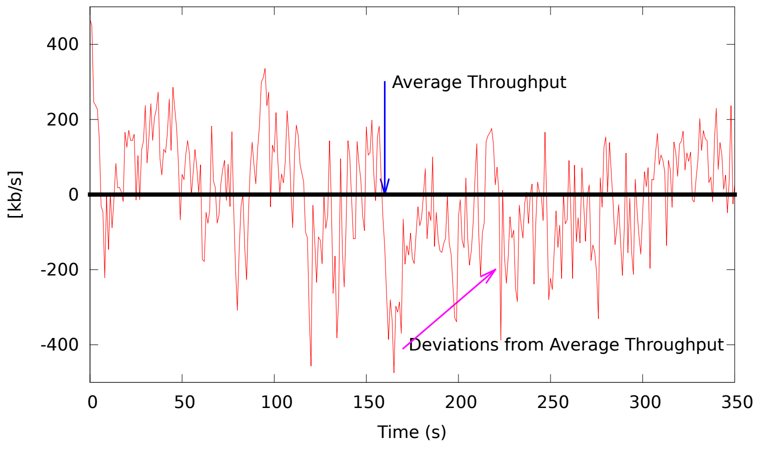


Figure 4.6: How throughput varies from the average over time in a typical 3G network.

maximum sustainable video rate, and the buffer level that will guaranty it, is to be called the optimum buffer level (B_{opt}).

Suppose that the change in the estimated throughput, calculated using Equation (4.5), is $\tau = c(t_i) - c(t_{i-1})$, and the rate at which τ changes as time passes is $\dot{\tau}$ [4]. To better understand the impact of throughput fluctuation ($\dot{\tau}$) on the system, measurements of various types of network transmission channels has been conducted. Figure 4.6 presents a throughput variation pattern of a 3G network that is observed, which is found to be a typical representation of the various transmission channels studied. It is worth emphasising when making a rate adaptation decision the only interesting issue with regard to the characteristic of the transmission channel is the *extent to which throughput deviations affect the system's chances in converging at the target video rate, in a stable manner.* For instance, if rate map, from Equation (4.10), allows for q_k, an ABR designer is only interested in the fraction of this video rate that the current capacity cannot meet. The relationship between $\dot{\tau}$ and the selected video rate can be modelled thus:

$$\dot{\tau} = \omega_t q \tag{4.13}$$

where ω_t is the link characteristics factor at time t after the start of the session,

[4]A negative $\dot{\tau}$ means the available bandwidth is drifting away from the average and the positive sign indicates a rise in the available capacity.

henceforth to be called system **robustness factor**. Without loss of generality, let us assume that $\dot{\tau}$ is the only exogenous component. Therefore, $z(t) = \hat{\tau}$ from Equation (4.1). The proposed framework mandates that where exogenous component exists, we have to use it as an adjustment factor in our model, therefore, to incorporate this requirement Equation (4.6) is rewritten as:

$$\frac{dq}{dB} = \alpha q \ln\left[\frac{q_{max}}{q}\right] - \omega_t q. \tag{4.14}$$

To solve Equation (4.14), it has to be reformulated into a form that will be easier to solve. Let us assume that $\beta = \ln(q_{max}) - \frac{\omega_t}{\alpha}$. By substituting β into Equation (4.14) the following equation is obtained:

$$\frac{dq}{dB} = \alpha q(\beta - \ln q) \tag{4.15}$$

To get the new rate map $y(B)$, which will henceforth be called Rate map 3, the differential equation in (4.15), using same condition as in the Section 4.4 is solved. That is, the streaming session starts with the minimum video quality level ($q = q_{min}$ and $B = B_{t_0}$):

$$y(B) = \mathrm{e}^{\left[\beta - \{\beta - \ln q\}\mathrm{e}^{-\alpha B}\right]} \tag{4.16}$$

4.5.1.1 Convergence

At the start of a streaming session, a client need not always start streaming with the minimum video rate, recall that a client receives the MPD before any chunk, hence the client can use this transaction to have some idea of what the system capacity is. Therefore, we now relax our assumption that an ABR has to start at q_{min}. Let us suppose that the client start with video rate $q_k \geqslant q_{min}$, provided that the available throughput is greater than the requested video rate $c(t_i) > q_k$, the buffer level grows. This allows a player using $y(B)$ for rate selection to progressively increases its video rate until it reaches q_s where $q_s \leqslant c(t_i)$, since Policy 2 constrained the client not request any video beyond the estimated capacity. The value of q_s from equation

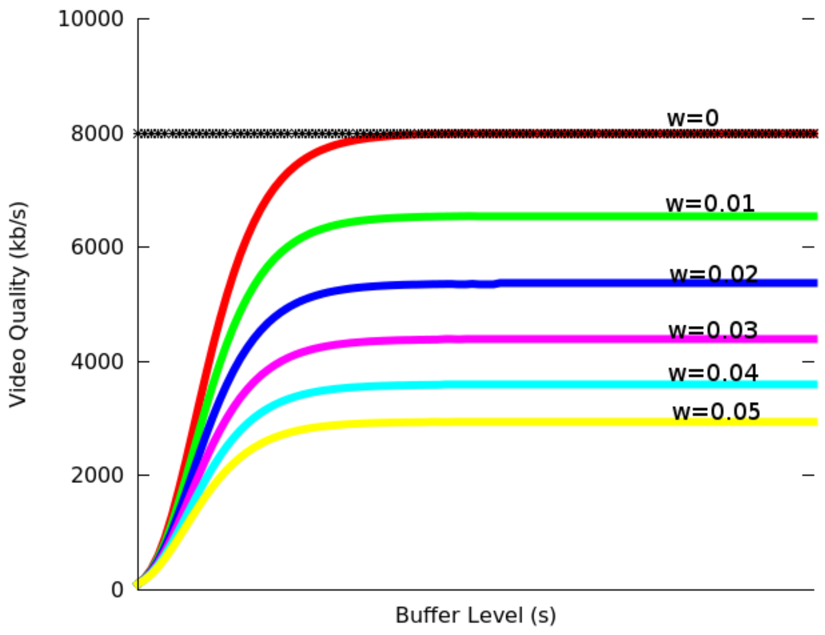


Figure 4.7: The impact of channel characteristics ω on the converged video rate q_s

(4.16) can be found by solving for the limit of Equation (4.16) as buffer tends to infinity:

$$q_s = \lim_{B \to \infty} y(B) = q_{max} e^{-\frac{\omega_t}{\alpha}} \tag{4.17}$$

Figure 4.7 presents a number plots of Equation (4.16) with different average values of ω. In the plots, $q_{max} = 8000kb/s$, $q_{min} = 100kb/s$, and $\alpha = 0.05$. As predicted by Equation (4.17) any increase in throughput deterioration results in the rise of the robustness constant, which causes a fall in the average throughput. Consequently, a drop in the value of q_s. This can be generalised as:

$$\forall \omega_t > 0 : q_{max} \geqslant q_{max} e^{-\frac{\omega_t}{\alpha}}. \tag{4.18}$$

4.5.1.2 Video Rebuffering

The Policy 3 required that only a rate map that guarantees the absence of unnecessary rebuffering events is employed for video rate selection decisions. Rebuffering occurs when the buffer is completely depleted, a distinct condition that may result in this rather unfortunate situation is when $c(t_i) \leqslant q_{min}$. In this situation, any video rate selected will result in the download rate being less the video rate. Hence, the buffer depletion rate will be faster than the refill rate, if this persists the buffer level becomes zero. However, whenever $c(t_i) \geqslant q_{min}$, as the $B \to 0$ the $y(B) \to q_{min}$, a point at which $y(B) = q_k$ will be reached, where $\{q_k \leqslant c(t_i) : \ q_{min} \leq q_k \leq q_s\}$. This results in $\dot{B}$ changing to positive, indicating to us that provided the available throughput is enough to cover at least the minimum rate the system will not rebuffer.

4.5.1.3 Stability

As in the previous case, the equilibrium points can be obtained by simply equating the Equation (4.14) to zero, which will result in two equilibria. The first is $q^* = 0$ and the second is $q^* = q_{max}\mathrm{e}^{-\frac{\omega_t}{\alpha}}$.

Just like in the idealised case, in Section 4.4, $q^* = 0$ is an unstable equilibrium. Therefore, we focus our attention on the second equilibrium point. When the system is perturbed near this equilibrium point, the system switches its video rate to q_k. When $\{q_k : 0 < q_k < q^*\}$, all values of $\ln(q_{max}/q_k)$ are positive, which means $\frac{dq}{db}$ is increasing, thus $q_k \to q^*$. However, when $\{q_k \geqslant q^*\}$, the all values of $\ln(q_{max}/q_k)$ are negative, which means $\frac{dq}{db}$ is decreasing, therefore $q_k \to q^*$. Hence, the second equilibrium point is asymptotically stable.

However, the second equilibrium point may keep changing depending on the impact of the exogenous component of the system, provided $c(t_i) = q_{max}\mathrm{e}^{-\frac{\omega_t}{\alpha}}$ the second equilibrium is locally asymptotically stable. This fact mandates the tracking of the robustness factor as the state of the system changes. Since the most optimal target is to have the video rate converged at the average throughput, that is, $q_s = c(t_i)$, the value of robustness factor that can guarantee the video rate equals the average throughput is certainly needed. This can be obtained by solving the equation (4.17), assuming $q_s = c(t_i)$:

$$\omega_t = -\alpha \ln\big(\frac{q_{max}}{c(t_i)}\big) \tag{4.19}$$

The equation tells us that the closer we are to the target video rate the less the impact of link characteristic is affecting the system.

4.5.2 Dynamics of User Experience II

As shown in Section 4.5.1.2, the Rate Map 2 is in compliance with the Policy 3. Therefore, when the model is used in a challenging environment two factors are likely to affect user experience[5]. First, a drop in the average video quality, since the converged video rate may be less than the maximum available video rate. Secondly, video rate fluctuation. While the first results in a reduced Delight Index (DI), the second makes it necessary to incorporate Annoyance Index (AI) when modelling user experience.

As the result of video rate change, the player changes its video quality level from θ_i to θ_k where $k \neq i$. Therefore, the amplitude of the switch represented by the perceptual change in video quality can be represented by:

$$SS_i = \mu|\theta_k - \theta_i| \tag{4.20}$$

However, Liu et al. [142] have shown that the impact of a switch on user experience depends on whether the video quality is being increased or decreased, with the increase in the video quality having a much smaller impact than the decrease in it. The parameter μ is introduced to take care of this fact when $k < i$ (i.e switch-down) $\mu = 1$ and when $k > i$, $\mu < 1$. In other words, when the video quality has reduced the value of SS_i remains as it is, but if video quality is increased only a fraction of it is used [6].

As network conditions vary, the value q_s fluctuates, such that not only will a user have to live with a switch but, also, a recurring one. Figure 4.8 shows the normalised result of subjective experiments conducted in [149] to access the impact

[5]Recall from Section 4.5.1.2 the system will not rebuffer, provided the available capacity is sufficient to sustain the minimum video rate.

[6]Find the value of μ requires an advance psychological study that is outside the scope of this thesis.

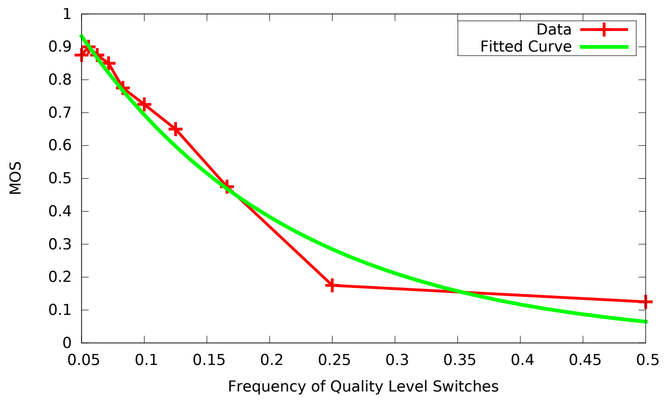


Figure 4.8: The impact of number of video quality switches on the user experience

of the frequency of video quality switches (f) on user experience. As can be seen, there is an increasing deterioration in user experience as the number of quality changes increase, which confirms our earlier argument, summarised by the Definition 4.2. In order to generalise the impact of the frequency of quality switch (IF), a curve fitting is conducted using a number of models, which found a simple exponential decaying function having the highest Pearson Correlation value (0.9076) (see the red line in Figure 4.8). Equation (4.21) represents the derived model:

$$IF = 1.254032\mathrm{e}^{-5.9262f} \tag{4.21}$$

where $f = \frac{N}{T}$, with N being the number video switches recorded within a shifting slide window of size T to be called the *perception window*. To get the actual depreciation in user experience, $SF = (1 - IF)$ is used. Therefore, when the video quality changes, a user suffers from both the deterioration as a result of the switch itself and also as a consequence of the increase in the switching frequency, hence the intensity of impact on a single switch will now be:

$$SI_i = SS_i(1 + SF) \tag{4.22}$$

Furthermore, in [164] it is shown that the impact of video quality distortion degrades by 70% after 20 seconds from the time it happens. Now, the cumulative effect of the video switch SI at any time t_i after the start of the session can be captured thus:

$$SI = SI_i \mathrm{e}^{-0.06019(t_i - t)}. \tag{4.23}$$

The following equation is used to compute the user annoyance that accumulates over time:

$$AI(t) = \frac{1}{N}\sum_{i=1}^{N} SI_i \tag{4.24}$$

By substituting Equation (4.12) and (4.24) into Equation (4.2) the cumulative user experience can be computed as thus:

$$UX = h\theta_{k-1} + (1-h)\theta_k - \frac{\lambda}{N}\sum_{i=1}^{N} SI_i \tag{4.25}$$

4.5.3 Discretisation of the Model

Up until now, it has been assumed that a player can choose any video between q_{min} and q_{max}. However, in reality, a HAS player is constrained to pick from a finite set of video rates. To reflect this fact, assumed that a player must only pick a valid video rate contained in an MPD. Suppose a player has received q_k, the ABR scheme may switch to any video rate $q_{k\pm n}$ (i.e. it can take n-steps). In this case, a switch from q_k can either be to q_{k-n} or q_{k+n} depending on whether the buffer is increasing or decreasing. This implies that provided the video rate return by the equation (4.16) is not a valid video rate the player will disregard it. This forces the player to stick with a constant video rate during a time interval. To calculate the size of the buffer interval needed before a particular video rate can be requested Equation (4.16) is expressed in terms of B:

$$B(q_k) = \frac{1}{\alpha}\ln\left[\frac{\beta - q_{min}}{\beta - q_k}\right] \tag{4.26}$$

4.6 Summary

In this chapter, the framework proposed in Chapter 3 is improved. In the new design, the evolving state of the user experience is made an active component of an ABR module. That is, any decision regarding a change in video rate is required to take into account the state of user satisfaction. To be able to model such a scenario adequately, a good understanding of the general dynamics of ABR module is required. Thus the first section of this chapter was dedicated to general system modelling.

From our understanding of the implementation-independent ABR dynamics, at both the system level and the user experience plane, a unified framework that maps the relationships between two planes was proposed. The proposed framework shows that there is a direct relationship between system input and the delight experienced by a user of a video streaming service. Furthermore, the framework argued that since a typical content provider does not influence the capacity of the last-mile and characteristics, network state not QoE should be considered as the exogenous component of an ABR system.

From the proposed framework, a model of an ABR scheme operating in an environment where the network capacity was both abundant and stable was developed. In this scenario, it was shown that the model needs not to take into account the influence of the adverse effect of the external factors. And it was also demonstrated that the model will always guarantee the highest level of satisfaction to a streaming user. After that, a more common scenario is modelled, in which the system capacity can both dwindle and fluctuate. Therefore, the model is required by necessity to take into account the impact of the exogenous components. Furthermore, since the user experience will certainly be affected, a model of the evolving state of QoE is constructed. This furnishes a designer of algorithms with all the information required to build a system that optimises QoE.

Chapter 5

Cooperative Streaming Service

5.1 Introduction

In the previous chapter, a framework that advocated the inclusion of the relevant QoE information in making video rate selection decision was presented. From the framework, some models that describe the relationships between the various components of ABR are derived. However, the models are not algorithms, but rather an elucidation of the system properties at any given time, which allows an ABR designer to better built a service that ensures a high user satisfaction. In this chapter, an algorithm built on top of the proposed models is presented, with the purpose of demonstrating how the framework can be used to build a practical service.

The chapter begins by presenting a case study of a typical streaming context that is both challenging and ubiquitous. Then we propose an algorithm, derived from the Framework 2 that is purposely designed. Even though it is entirely possible to use the framework for developing either opportunistic or cooperative algorithm, in this thesis, a cooperative streaming service is presented. The algorithm is, for the sake of brevity, divided into two parts. The first part collects the necessary contextual information, and appropriately configure the model parameters. The second part uses the gathered data to make the rate selection decision. It should be noted that the second part of the algorithm, that is Algorithm 3, can be used as a standalone opportunistic ABR.

5.2 Case Study

With the current burgeoning access to mobile devices and their increasing use for video streaming, it is natural for content providers to be interested in the performance of a video streaming service in a wireless mobile context. This section describes the environment in which the proposed algorithms are expected to be used in.

A scenario is assumed where the streaming service is used on board a moving vehicle, such as a car or a train. It is now becoming increasingly common for transportation companies to provide either free or paid shared commercial Internet access to their passengers. It is expected that some of the users, of the proposed video streaming service, may opt to use such services, while other may decide to use their independent Internet connections. While streaming on board a vehicle, fluctuation in the channel capacity is bound to be a common occurrence. In fact, it is not unusual to pass through areas where the strength of Internet connectivity is either low or even completely unavailable, possibly for a prolonged period. Notwithstanding this situation, as extensively discussed in the Section 2.8, users like a streaming service with:

- limited number of video stalling events,
- low video quality oscillation,
- good perceptual video quality,
- low data download.

Furthermore, for the category of users using a shared connection, fairness is important, not only in terms of system-level metrics (such as network utilisation and average video rate) but also with regards to the QoE metrics (such as visual quality of the video, video quality fluctuation, and the amount of the data received). In this chapter, we use the volume of the downloaded data as a proxy of the cost and the energy expenditure of a streaming service.

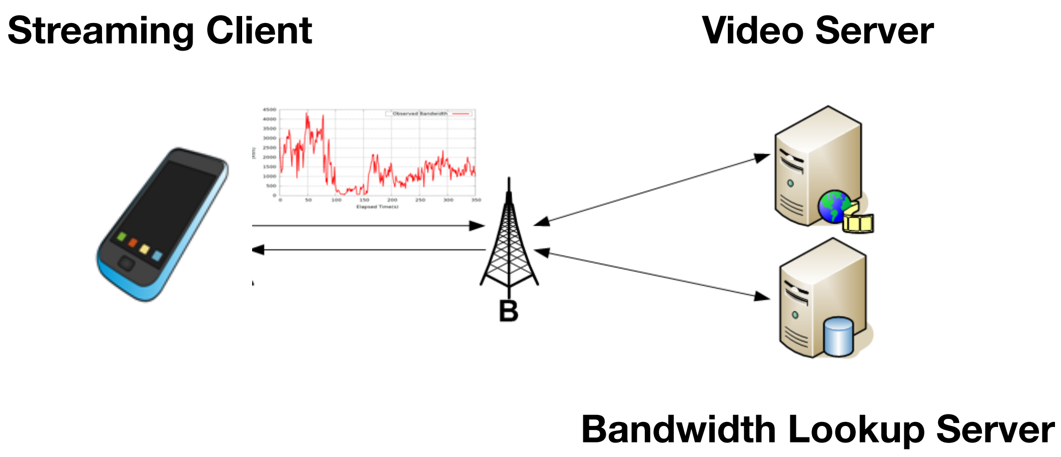


Figure 5.1: Video streaming setting

5.3 Cooperative Algorithm

If a context rapidly changes, the TCP throughput fluctuates or may go below the threshold needed to even stream at the minimum available video rate safely. Figure 5.1 presents a visual representation of the context described in the previous section. As can be seen, the connection between the streaming client and the base station B continuously changes, we assume that this happens because the connection between the two nodes is constantly changing (recall that the streaming client is used in a moving vehicle). In this kind of environment, opportunistic algorithms, like the throughput and buffer-based protocols discussed in Chapter 3 will find it difficult to know if their target is realistic or not, hence will continuously target the highest video rate advertised in the MPD. This is because such schemes are usually reactive, that is, they make decision based on the recently collected data. However, historical estimates are often not the actual representation of the future. Therefore, continuously targeting the advertised highest video rate, based on unreliable information, may result in an aggressive behaviour especially in an uncertain context like a wireless environment.

Since an opportunistic scheme is by nature is 'myopic', the algorithm proposed next, will be *cooperative*. This means each streaming client will cooperate with others to better improve the information obtained by all. The proposed algorithm is made up of two parts. The first part collects the relevant information and the makes it available to the participating streaming clients. The second part is a standalone algorithm that a client uses, together with the information avails to it by the first part of the algorithm, to decide on the video rate to request.

5.3.1 First Part

It is assumed that a content provider periodically collects simple information about the state of streaming devices, such as geographical location information, throughput estimate of the individual clients, the number of players streaming in a given area, and the type of network connection used for streaming (e.g. WiFi). The information is then stored at a server, to be henceforth called *Bandwidth Lookup Server*, that is accessible to all users. The Bandwidth Lookup Server shall periodically create a record containing the collected data in aggregated format (e.g. the average and standard deviation of the throughput in a given location).

On arrival at a location, a client intending to use the service queries the bandwidth lookup server for the average and standard deviation of the available system capacity along the vehicle's route. We assume that a path prediction algorithm that allows both the client and the server to agree on the trip route is available. For train but also car it is not difficult to predict the route because their paths are often predefined and do not change. After that, the client is expected to request bandwidth information from the server regarding its current location and some distance along its route. On the receipt of the client's request, the server first checks the number of active sessions in the area and then responds with the appropriate information that matches the client's current context. After receiving a successful response from a server, the proposed algorithm as represented in Algorithm 2 is activated.

Algorithm 2 begins by taking the following list of inputs: current q_{max}, which may be the one provided by the MPD or the one previously computed; the current throughput as estimated by the client; and the trace file obtained from the bandwidth

Algorithm 2: Part 1 of Algorithm

```
input  : Trace data, $q_{max}^{i-1}$, $c(t)$
output: $q_{max}^{i}$: Next maximum video rate
avgT: Average throughput of the trace
stdT: Standard deviation of the trace throughput
if $c(t)$=:0 then
    $c(t) = avgT$
end
$\omega_c = \omega[(c(t), q_{max}^{i-1}]$
$\omega_f = \omega[avgT, q_{max}^{i-1}]$
$q_s^c = q_s(\omega_c)$
$q_s^f = q_s(\omega_f)$
if $Pr(avgT, stdT, q_s^f) \geqslant Pr(avgT, stdT, q_s^c)$ then
    $q_{max}^{i} = q_s^f$
else
    $q_{max}^{i} = q_s^c$
end
```

lookup database. The output of the algorithm will be the maximum video rate that the system should target. It is worth noting that at the start of a streaming session or when a context has drastically changed, such as after exiting a tunnel or crossing a river, the algorithm may not have any valid estimate of the system capacity. In this case, the client shall use the throughput estimate it receives from the bandwidth database [1]. Given this information, the algorithm computes two robustness factors using Equation (4.19). The first, called *current robustness factor*, is calculated using the client's estimate of the available resource. The second factor is computed from an average bandwidth derived from the received trace, hence called *future robustness factor*, because it stems from the data points of the area yet to be visited by the client running the algorithm. For the computation of both robustness factors, the current q_{max} is used [2]. Using these two robustness factors, and the current estimated throughput, two target video rates (q_s^c) and q_s^f are computed. The former tells the system if the current situation persists q_s^c is the most optimal target while the latter

[1]However, it is arguable that the download of MPD can provide an estimate of the system capacity, while this can be used, because the results of measurements conducted showed that the size the MDP files used in the course of the thesis are very small.

[2]It is worth emphasising that at the start of streaming, q_{max} is the one provided by MPD, and $c(t)$ is derived from the trace data queried from a server.

tells the system given what is known about the future q_s^f is the best choice.

Using a representative distribution, the probabilities of achieving the two targeted video rates (q_s^c and q_s^c) are calculated. The target with the highest probability is assumed to be the new q_{max}. This value will now be used, until the life span of the trace expires, in calculating $y(B)$. The client will be able to use Algorithm 3 based on the recomputed $y(B)$ to select the appropriate video rate. This gives the algorithm an ability to look ahead, and hence better plan for the future degradation in channel state.

5.3.2 Second Part:

The algorithm is invoked at the start of a video streaming session, and then continuously called after each successful download of a requested chunk. Each call of this part of the algorithm takes the following arguments as input: video rate of the previously downloaded segment, current buffer level, robustness factors, switching step [3], the size of the off-period, and the recently computed q_{max} (using Algorithm 2). The output is the video rate.

At $t = 0$ it is assumed that the buffer is empty. Therefore the algorithm starts by requesting the lowest video rate q_{min}. Thereafter, if the current buffer level is equal or greater than the space needed to switch to the next video level, as described at Section 4.5.3, that is, $B(q_{k+s}) - B_t \leqslant 0$. And then, the algorithm checks if the quality of the video it is switching to enhances user experience. This is done by making sure that the next video quality is more than the current one by at least $\theta_k|\omega(c(t_k))|$. In other words, before a video quality is changed, the new level has to be high enough to withstand the current throughput fluctuation. This increasingly dampens the video quality switching rate of the algorithm as the TCP throughput fluctuation rises.

However, if the buffer space is not sufficient to enable video rate increase, the ABR checks if $B_t - B(q_{k-s}) \leqslant 0$, that is, if the buffer level is below the threshold that

[3]This is the number of the video levels to skip when switching video rate, i.e. that number of video levels between the current rate k and the next video rate, for example $s = 1$ means move to the $k + 1$.

Algorithm 3: Part 2 of the Algorithm

```
input : $q_{max}$
        $q_k$
        $B_k$ $\{B_k : 0 < B_k \leq B_{max}\}$
        $\omega(c(t_k))_{k=1,\ldots t_e}$
        $s$ (Switching steps)
        $V$ (Off period)
output: $q_{next}$
        $B_{delay}$
if $B_t := 0$ then
    $B_{delay} = 0$
    $q_{next} = q_{min}$
else
    if $q_k =: q_{min}$ then
        $q_{next} = q_{min}$
    else if $q_k =: q_{max}$ then
        $q_{next} = q_{max}$
        $B_{delay} = V$
    else if $B(q_{k+s}) - B_t \leqslant 0$ then
        if $\Theta(y(B_t)) \leqslant \theta_k(1 + |\omega(c(t_k))|)$ then
            $q_{next} = q_k$
        else if $k + s \leqslant n + 1$ then
            $q_{next} := q_{k+s}$
        else
            $q_{next} := q_{max}$
        end
    else if $B_t - B(q_{k-s}) \leqslant 0$ then
        if $k - s \leqslant 0$ then
            $q_{next} = qmin$
        else
            $q_{next} = q_{k-s}$
        end
    else
        $qnext = q_k$
    end
end
```

can sustain the next lower video rate. And provided that the current video rate is not the minimum available, the video rate is switched-down. Otherwise, the video rate is maintained at its current level. Note, whenever the buffer is full the system ceases sending requests to the server for a time equivalent to the chunk size as dictated by the Policy 1. That is at least one chunk has to be played before a new request is dispatched.

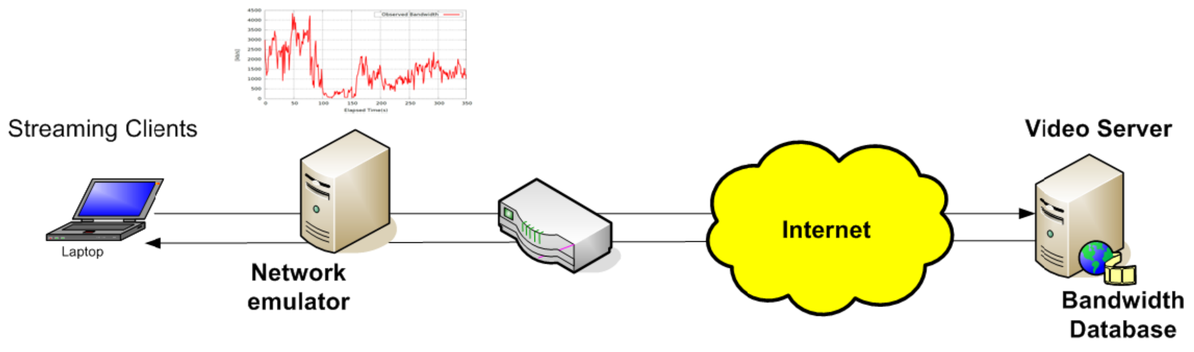


Figure 5.2: Experimental Set-up

5.4 Experimental Set-up

The experimental set-up used in emulating the case study discussed at Section5.2 is shown in Figure 5.2. For the first set of experiments, only a client is used in streaming video. This scenario emulates when a client is streaming with its private connection. In the second sets of experiments, some clients compete for the available capacity to emulate when clients are using a shared connectivity. All streaming clients are connected to a rate limiting node, which sits in-between the client and the Internet gateway. The rate limiter uses *Dummynet* to emulate the bandwidth of the different scenarios investigated. Two (2) bandwidth patterns are emulated:

1. In the first campaign, a client starts streaming in a context with high capacity that gradually degrades, and remains in the state of limited capacity for considerable amount time, after which the available system capacity recovers.

2. In the second campaign, the client begins with limited capacity after some time the situation improves.

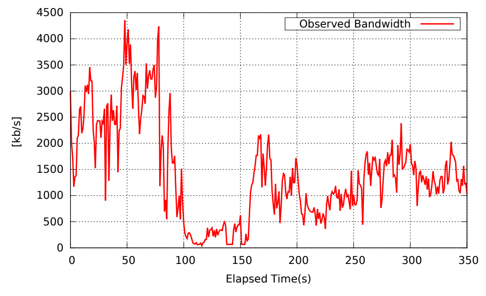
The first scenario is meant to emulate a situation when a client starts streaming before going into a capacity constrained context and then exits later. The second scenario emulates a situation where a client starts streaming from within a tunnel and continues after exiting. To make the tests as realistic as possible, a bandwidth trace presented in [2] for emulating the mobile scenarios is used. The chosen traces contain the measurement campaigns of ‘popular commute routes in and around Oslo

(Norway)'. For the first campaign, a trace collected from streaming in a car from Aarnes to Elverum is used, shown in Figure 5.3(a). While Figure 5.3(b) shows the second campaign, in which a trace collected while streaming on board a train from Oslo to Vestby is used. In each case, only the first $400s$ is used. To emulate a crowd-sourced data, noise is added to each of the traces and then stored at the bandwidth lookup server. This ensures that more realistic data is obtained. To do this, each entry in the trace is replaced by a randomly generated number from a Gaussian distribution with an average equal to the entry and the standard deviation of 20%. For the competing scenario, the entry is first divided by the number of competing clients, before the noise is added.

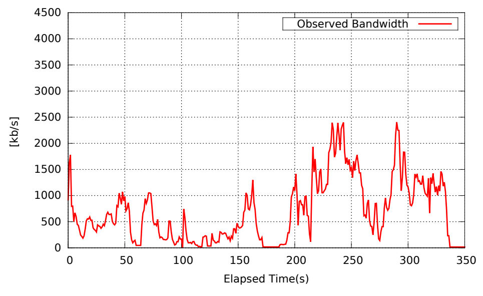
The web server hosts the MPD and the Big Buck Bunny video dataset [159], which is encoded in VBR. The web server is also co-located with the bandwidth lookup server storing the crowd-sourced data. All node run Ubuntu 12.04.2 LTS with 3.8 kernel. The host that runs the player also hosts *tcpdump* and *lsof*.

The performance of the proposed algorithm is benchmarked against the buffer-based player and the throughput-based player used earlier in Chapter 3. All the players are implemented in Python, and use the Request package for HTTP request-response transactions. $B_{max} = 40s$ is used throughout the experimentation. For the buffer-based player, one-third of the buffer space is reserved for the 'reservoir'. And for the throughput-based player, the same configuration as used in [22] is employed. For the player running the proposed algorithm $\alpha = 0.05$ is used.

To simulate the motion of a vehicle, after receiving a queried trace data, the client first uses the Geopy package [165], a Python geo coding toolbox, to calculate the distance between the first and the last points (GPS points) in the trace. Then generates a random speed from a normal distribution with $50kmph$ mean, and $20kmph$ standard deviation combining the two parameters (speed and distance) to derive the time interval between subsequent database queries. Each experiment is conducted ten times, and the average result is used. When more than one player is used, the test is conducted on the same machine for maximum portability.



(a) The recorded bandwidth along Aarnes to Elverum as observed from a car.



(b) The recorded bandwidth along Oslo to Vestby as observed from a train.

Figure 5.3: The first 400s of the 3G bandwidth measurements from the trace file [2].

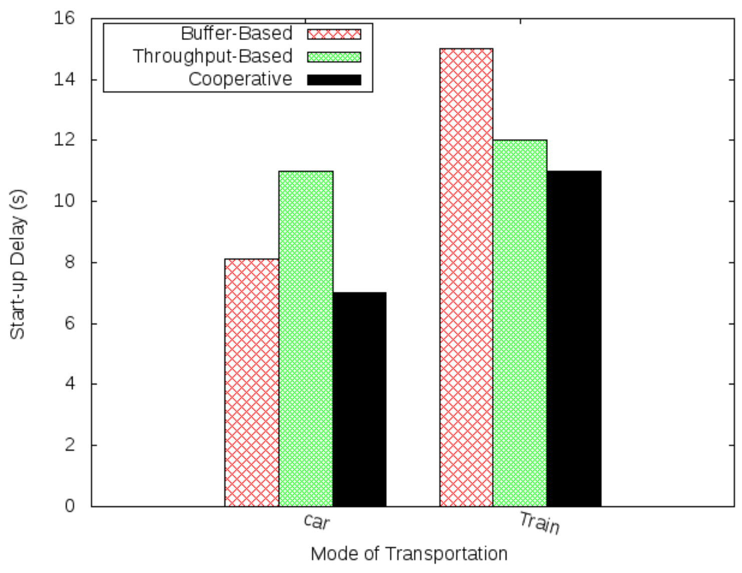


Figure 5.4: Start-up delay

5.5 Results

This section discusses the result of the various test-bed experiments conducted when the player does and does not compete with other players.

5.5.1 Scenario 1: Player Using All the Available Channel

In this set of the experimentation, a player streams video while commuting either in a car or on the train, each time streaming a video of 480P resolution with the minimum of 100kb/s and the maximum of 4500kb/s video rate.

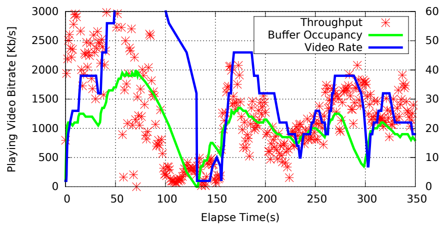
5.5.1.1 Start-up Delay

Figure 5.4 presents the start-up latency experienced by a user streaming video while on board a car or a train. Recall that the from trace the car has a higher starting capacity than the trace from the train, which allows the client to request commensurately high video rate. This explains why across all the players the user experiences longer startup delay when streaming on board a train. However, in both mode of transportation the player running the proposed cooperative algorithm experiences the least startup delay as compared to the baseline players, with the buffer-based player suffering the highest delay of up to $28s$ while streaming on the train.

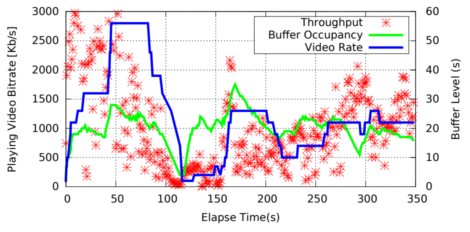
5.5.1.2 Evaluation of the Impact of Network Fluctuations on Video Rate and Buffer

Figure 5.5 and 5.6 show the detail of the impact of different patterns of throughput change on the behaviour of all the players used in the evaluation. The first thing to note is that when the baseline players sense that the bandwidth is abundant, as shown in Figure 5.5(a) and 5.5(b), both players aggressively ramp-up the video rate of their requests. But suddenly the bandwidth drops forcing the both players to reduce the video rate of their download. While the throughput-based player can do that without depleting its buffer, due to the high amplitude switch, which is known to affect user experience. The buffer-based player is not capable of dropping its video rate in time to avoid buffer depletion. Consequently, a video freeze of about $7s$ is experienced. However, the proposed cooperative algorithm by using information about the possible future is able to be modest in its download, and wisely pre-load content when the network capacity allows until it fills the buffer to the maximum. This prevents the cooperative algorithm from depleting its buffer quickly or having to suffer from a high amplitude switch.

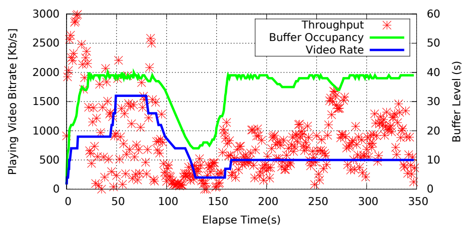
Furthermore, when the system recovers both baseline players continue with their aggressive video rate change. However, this time, the TCP throughput incessantly fluctuates, forcing the players to change their video rate persistently. As can be ob-



(a) Buffer-based Player

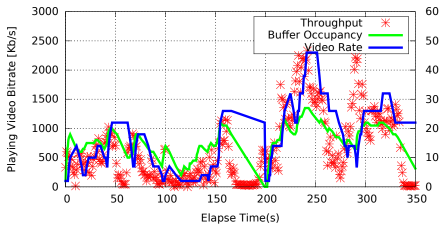


(b) Throughput-based Player

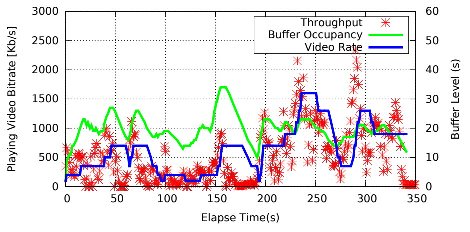


(c) Cooperative Player

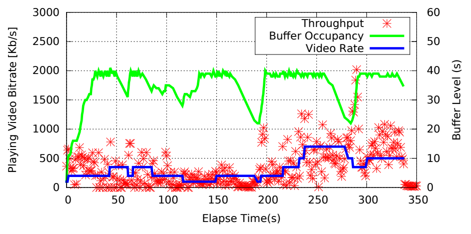
Figure 5.5: Throughput change and its consequent impact of the dynamics of video rate and buffer state, when streaming while commuting in car.



(a) Buffer-based Player



(b) Throughput-based Player



(c) Cooperative Player

Figure 5.6: Throughput change and its consequent impact of the dynamics of video rate and buffer state, when streaming while commuting in Train

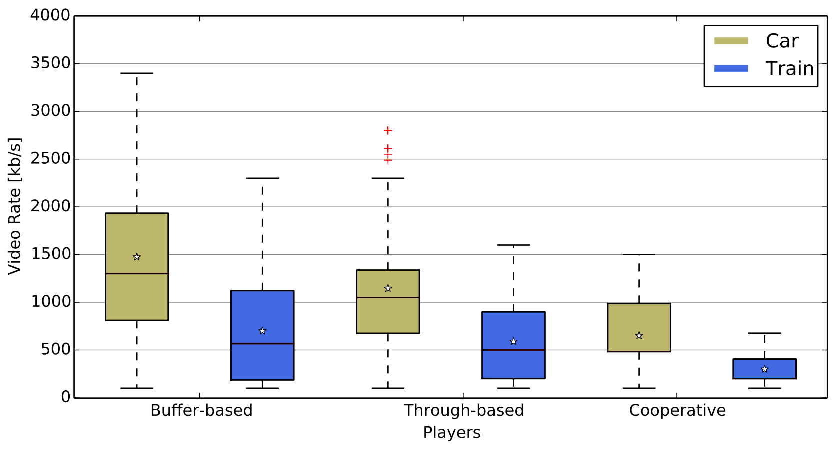


Figure 5.7: The distribution of video rate when streaming on board car and train.

served from Figure 5.7, this results in the baseline players having high mean value and variability in video rate, with most of the variation towards the high end. However, by incorporating the information about the likely future variability in throughput, and ensuring that a video rate is only switched-up if it will result in an increase in the user satisfaction, the proposed player is able to stabilises its download rate, hence record less variability in its video rate (see Figure 5.7).

Similar behaviour is observed when a player streams on board a train. With the baseline players trying to tightly fit their requested video rate to the available capacity suffering high fluctuation in video rate, and video freeze in case of buffer-based player (see Figure 5.6(a), 5.6(b) and 5.7). Remarkably, as can be seen in Figure 5.6(c), the proposed cooperative player is not only able to moderate its download rates to better fit both the prevailing and the likely future network state but is also able to provide a more consistence download rate.

Figure 5.8 presents a plot of the distribution of the data delivered to each player, as a percentage of the combined total data delivered to all the clients. As can be seen, regardless of the streaming context, the baseline players significantly download more data than the proposed player. In both scenarios, the player running the throughput-based algorithm receives on average 33% of the entire data downloaded by all the

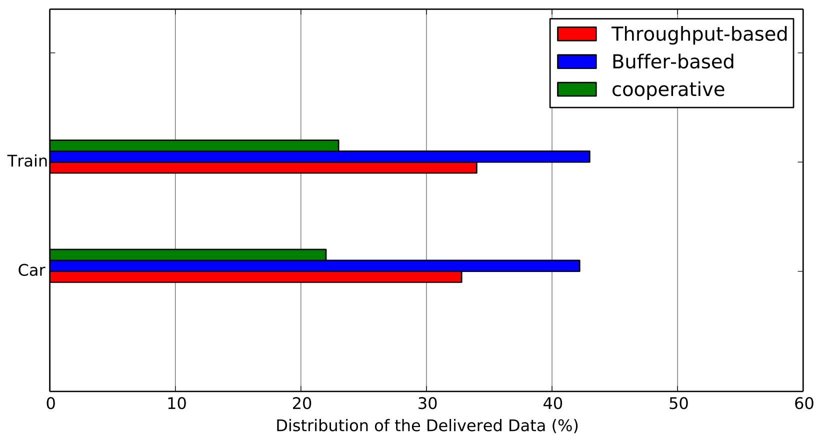


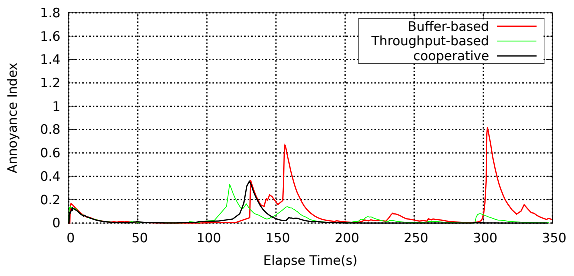
Figure 5.8: The distribution of the delivered video data, as percentage of the total data consumed by all the players, when streaming on board car and train.

clients, while the buffer-based player downloads 43%. However, the cooperative player receives 22% of the whole data, which implies that the proposed player receives 45% and 86% less data than the throughput-based and buffer-based players, respectively. The consequence of this large reduction in data download is that the cooperative service is going to save users more money, and also, reduce drain in battery power.

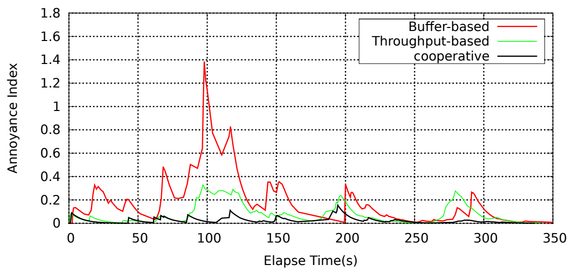
5.5.1.3 Evaluation of Network Fluctuations on Video Quality

So far, how the players under study behave in the presence of different patterns of throughput change is seen. Here, a look is taken at how this behaviour affects user experience. As shown in Equation (4.25), the perceived user experience is the video quality of the downloaded chunk less the annoyance suffers as the result of video quality fluctuation.

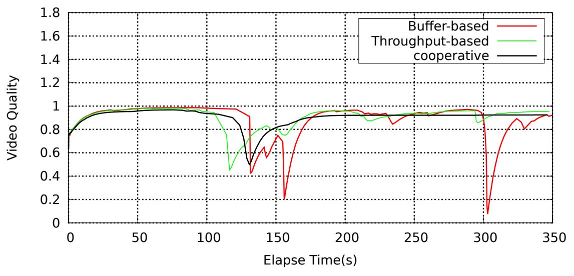
Figure 5.9(a) and 5.9(b) show the detail of the extent of annoyance that the user suffers as the result of video rate fluctuation (shown in Figure 5.7). When streaming onboard either a car or a train, the user is most annoyed with the buffer-based players, followed by the throughput-based player. However, seeing only an instant of a slight increase in annoyance while streaming with the proposed cooperative player on board



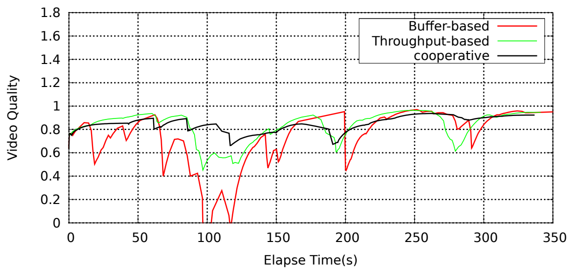
(a)



(b)



(c)



(d)

Figure 5.9: The impact video rate fluctuation on the perceived user experience. Annoyance suffers while streaming in Car (a) and (b) while train. (c) is the user experience while streaming in car and (d) in train.

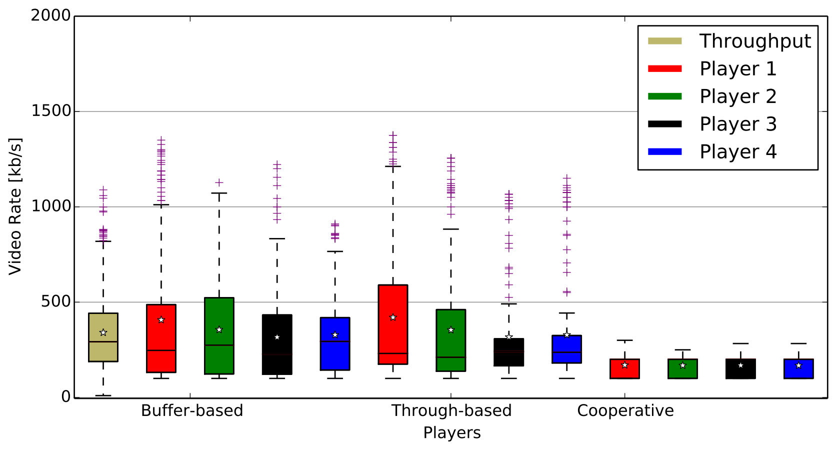


Figure 5.10: The distribution of the video rate recorded by each player as it competes with ours on board a car and a train.

a car. It is worth noting that even for this one instant the proposed player is not able to prevent it because the system capacity has gone below what can even sustain the lowest available video rate. Remarkably, not even an instant of annoyance is recorded while streaming with the proposed player on board a train. Consequently, the overall user experience (see Figure 5.9(c) and 5.9(d)) is lowest when the buffer-based player is used to stream video in the both investigated scenarios. And the highest level of satisfaction is achieved when the proposed player is used, also worth recalling, is that this is achieved with less data download.

5.5.2 Scenario 2: Players Compete for the Available Capacity

When players compete for the available bandwidth at the bottleneck link, extreme fluctuations in video rate can be observed, which as we have seen has a negative impact on the user experience. Even worse, when the network capacity is low, an increase in video stalls may be the rule rather than exception. To study the impact of competition, in the scarce resource, on the proposed cooperative algorithm, four players are used to stream video via the same bottleneck link. The link emulates the bandwidth of a moving car, as discussed in section 5.4. The players were randomly started, with interval of the starting time between players normally distributed with a mean of eight seconds ($8s$), and the standard deviation of four seconds ($4s$).

5.5.2.1 Evaluation of the Impact of Competition on Video Rate and Delivered Data.

The distribution of the video rate of all the players as they compete for the available bandwidth is presented in Figure 5.10. The first box-plot, in dark-khaki colour, shows the ideal distribution of the fair allocation of bandwidth to each of the competing client, which is derived by only dividing the bandwidth of the trace, used in emulating the network, by the total number of the competing players. As can be observed, this distribution exhibits a high viability, and worse many outliers (also see 5.3(a) for more detail). Tightly matching this pattern may not necessarily be what enhances the user experience.

As can be seen, the buffer-based player made the best attempt, among the players under study, to match its video rates to the ideal distribution. The average video rate of all the buffer-based players is very close to each other, and the ideal allocation. Furthermore, there is a clear variation in video rate, both among the players and concerning the ideal distribution, in the case of the throughput-based players resulting in a marked unfairness in mean video rate. However, the proposed cooperative players show a high level of fairness in the distribution of video rate, with all players exhibiting similar performance. Nonetheless, they recorded the lowest average video rate.

Since all the competing players are exactly alike and are operating in a similar context, it is reasonable to expect that the data downloaded by each of the participating players to be reasonably close. Figure 5.11 shows the ratio of the data downloaded by each player to the total data downloaded by the set of the competing players using a similar algorithm.

It can be seen that the players running the proposed cooperative algorithm download an almost equal amount of data, with each player receiving a share that ranges

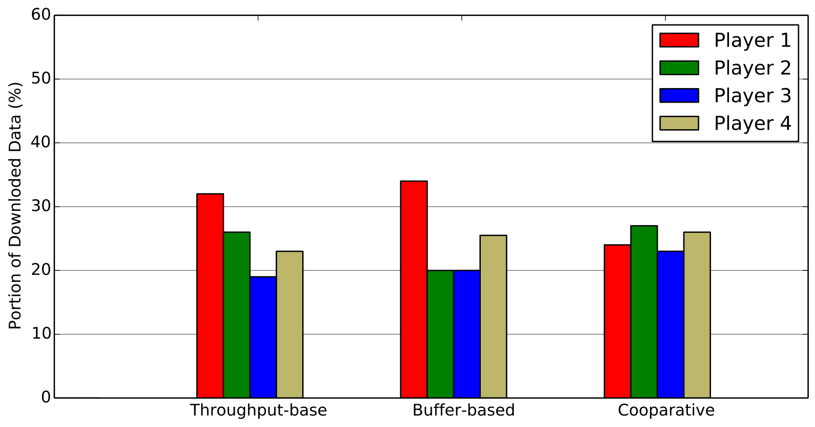


Figure 5.11: Link utilisation of the competing clients.

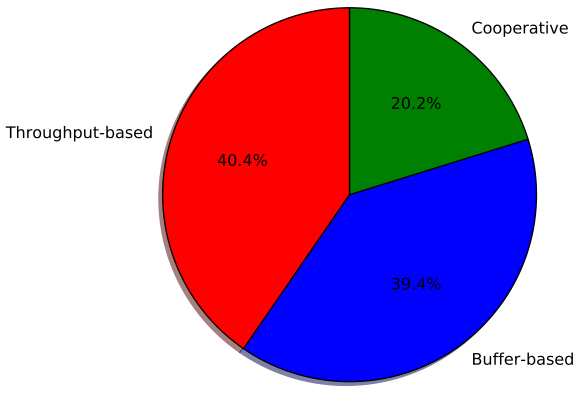


Figure 5.12: The distribution of the delivered video data, as percentage of the total data consumed by all the players, when players compete for the available bandwidth.

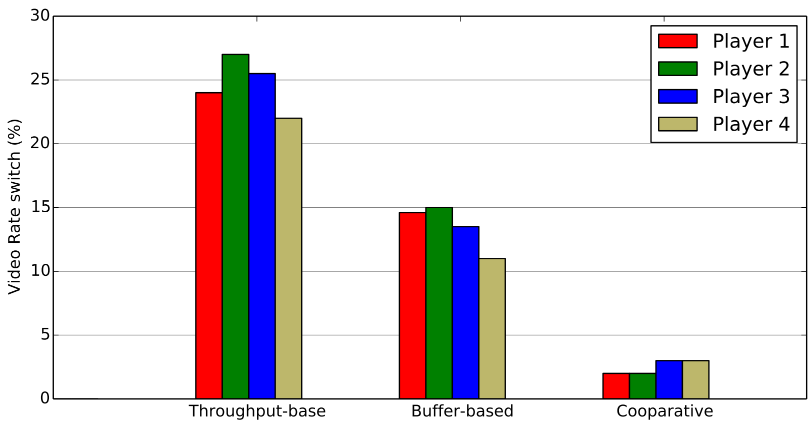


Figure 5.13: Stability of video rate when players compete for the available bandwidth.

from 23% to 26%. However, the distribution of the delivered data amongst the players running the baseline algorithms markedly varies. When the throughput-based algorithm is used, Player 1 received 33% of the downloaded data while Player 3 received 18%, in other words, Player 1 downloaded about 1.83x more data than player 3. A similar pattern can be observed when the buffer-based players compete. This level of unfairness may result in some players paying more than others for a similar service.

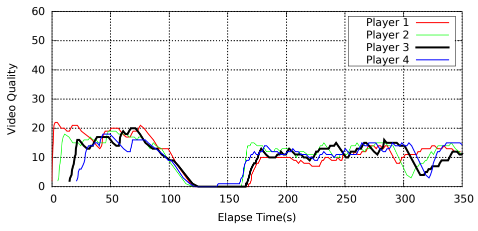
Now, the distribution of the delivered data amongst the algorithms under study is considered. First, the amount of the downloaded data of a group of players running a particular algorithm is summed-up, and the ratio of the bundled total to the combined total of the amount of data downloaded by all of the players is computed. From Figure 5.12, it can be seen that both baseline players download, approximately, 40% each of the combined total data. Simply put, they each receive twice the amount of the data delivered to the cooperative players, which receives a total of 20% of downloaded data.

While it may be desirable to closely fit the bitrate of the requested video chunks to the bandwidth when the bandwidth is stable, doing so when the bandwidth fluctuates may not be desirable. Figure 5.13 shows the detail of the extent of the video rate fluctuation in all the players investigated. When the throughput-based algorithm is employed, the players suffered the highest variability in video rate, with instability among the players ranging between 23% to 28%. The instability of the players drops to between 13% to 15% when the buffer-based algorithm is employed. The cooperative player shows this need not to be necessarily the case. It reduces the instability to less than 3%. Recall from Figure 5.10 that the baseline players tightly fit their video bitrate to the ideal fair allocation, the consequence of this over-fitting is, as just seen, incessant video rate switch.

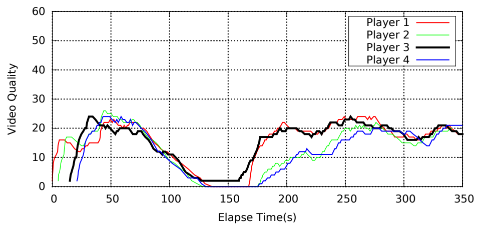
5.5.2.2 Evaluation of the Impact of Competition on Buffer

Figure 5.14 gives the details of buffer state changes as the four players compete for the available bandwidth. It can be seen, both baseline players started by gradually filling their buffer, in both situations, they try to maintain the buffer level at about 20s. However, at around 80 seconds from the start of the streaming session, the buffer begins to rapidly deplete until it becomes empty. As shown in Figure 5.3(a), this is the same time when the available bandwidth sharply drops to a level that is insufficient for the four players even safely to download the minimum video rate. The situation remains like this until when the bandwidth recovers to the level that allows all the players to resume the download. Table 5.1 presents the summary of what happened. Even though the video freeze happened only ones in all the baseline players, it lasts for a prolonged period. For the throughput-based player, the stall period ranges from 12s to 26s. For the buffer-based, the stall period significantly varies, while player 2 and 3 have a stall period of 24s, player 1 and 3 record 11s and 2s, respectively. It is worth noting that players that suffer the longest period of video freeze are the same that in Figure 5.10 were most aggressive with video rate increase.

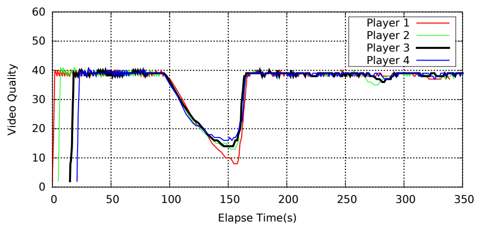
However, a different result is recorded when the proposed cooperative algorithm is used. From the very beginning of the streaming session, the cooperative player receives information about the future state of the TCP throughput. Additionally, it uses the information to assess the stability of the available bandwidth. This allows the player to concentrate on buffer replenishment. As can be seen in Figure 5.14(c) the player quickly filled its buffer to maximum (40s). This excess content is used to



(a)



(b)



(c)

Figure 5.14: The impact players competition on the buffer state changes (a) Buffer-based, (b) Throughput-based, (c) Cooperative player.

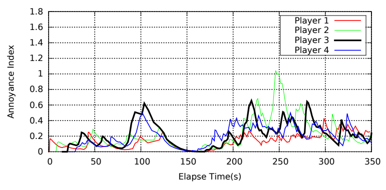
Table 5.1: Summary of the Various effort to Improve the Throughput Estimation

Dataset	Buffer-based		Throughput-based		Cooperative	
	Events	Duration (s)	Events	Duration (s)	Events	Duration (s)
Player 2	1	24	1	24	0	0
Player 1	1	26	1	11	0	0
Player 3	1	22	1	2	0	0
Player 4	1	12	1	24	0	0

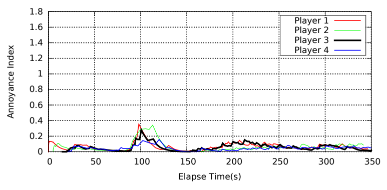
absorb the rapid fall in available bandwidth, which successfully prevents the video freeze.

5.5.2.3 Evaluation of the Impact Competition on Video Quality

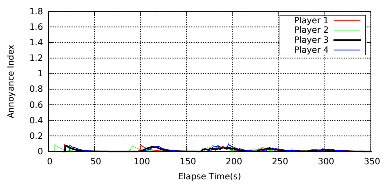
Finally, the impact of the findings in the previous section on user experience is discussed. Figure 5.15 shows the details of the impact of video players competition on the user experience, with Figure 5.15(a)-5.15(c) showing the annoyance suffer by the users as the result of the competition induced video quality fluctuations. Clearly, the



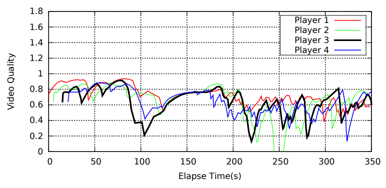
(a)



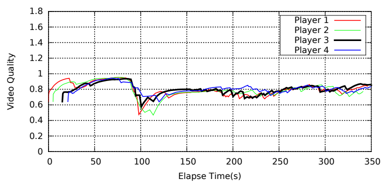
(b)



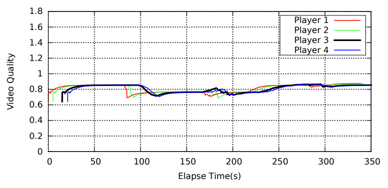
(c)



(d)



(e)



(f)

Figure 5.15: The impact players competition on the perceived user experience. Annoyance suffers while streaming in Car (a) and (b) while train. (c) is the user experience while streaming in car and (d) in train.

users streaming video using the buffer-based players experience the highest annoyance, followed by those using the throughput-based players, this is consistent with Figure 5.13 that showed the highest video rate fluctuations in buffer-based players. Consequently, the buffer-based players, as can be seen in Figure 5.15(d), not only experience the highest fluctuating video quality but also one that keeps on dropping to a level that is lower than what the user could have experienced if it were to stream at the lowest available video rate consistently. Hence, has least satisfactory user experience as shown in Figure 5.15(d). However, the players running the cooperative algorithm suffer virtually no annoyance (see Figure 5.15(c)), recall they recorded video rate instability rate of just 3%, hence, even though it recorded the lowest average video rate per player, it achieves the highest video quality.

5.6 Summary

In the previous chapter, a framework that incorporates the QoE metrics into video rate selection decision making has developed, from which several models that capture the relationship between the system level metrics, such as throughput and buffer occupancy, and the user experience are derived. In this chapter, an algorithm built on top of the models, to demonstrate how the models can be used in practice, was developed.

The proposed algorithm is cooperative, that is, all participating clients contribute in ensuring the effectiveness of the delivered service. Each streaming client continuously sends its network state information and the geographical location to a server. Before making any video rate decision, a client requests previous information about the network state along its current route. This allows the cooperative algorithm to ensures that (a) a player changes its video rate only if it will actually enhance the video quality perceived by a user, and (b) any current rise in video rate will not cause future degradation in the user experience. The result of the experimental evaluation conducted shows that the cooperative service improves the both the magnitude and the stability video quality, even when requesting lower video rate; eliminate unnecessary rebuffering, and cut down the amount of data download. Hence increases the overall user satisfaction when a player is using either a shared or dedicated transmission channel.

Chapter 6

Discussion

6.1 Introduction

In the previous chapters, this thesis has presented several frameworks, models and algorithms. More concretely, Chapter 3 starts by defining the valid states that a typical adaptive video streaming service should be in at any given time. Subsequently, the chapter presents a QoE-aware video rate map that governs the only allowable video quality evolution path. To demonstrate the impact that the scheme can bring to bear some state-of-the-art adaptive streaming algorithms are modified to work with the proposed models. Chapter 4 starts by improving the rate map introduced in the previous chapter and on top of this it models the dynamics of the video quality change at both system and user experience levels. In Chapter 5, we present a novel cooperative adaptive streaming service. Finally, the algorithm is implemented and evaluated to ascertain the effectiveness of the proposed scheme.

This chapter will bring together the different aspects of the research and reflects on the technical contributions of the previous chapters. It begins with a section that presents a detailed discussion of how the proposed frameworks and models relate and help in validating the hypothesis. The following section discusses how the research can be adopted and deployed in practice. This is followed by an in-depth analysis of some of the possible obstacles that we envisaged could be encountered in using the work in practice. Next, we review how the proposed schemes complement and differ from the existing work. The chapter concludes with a presentation of some of the shortcomings of the research together with some recommendations on the possible way forward.

6.2 View from the Top

In Chapter 1, we hypothesised that incorporating, at the algorithm design phase, a systematic model of the relationship between the various ABR components can help in improving the user Quality of Experience. After this, we set forth some goals that can assist in this regard. In this section, we revisit the work presented to assess the extent to which it helps in validating the proposed hypothesis.

6.2.1 Objective I

In this thesis, we take a system view of the adaptive bitrate selection module. We considered it to be a system made up of self-contained and interconnected functional components. Typically, the behaviour of an ABR module evolves because both its inputs and actions change in response to the change in the operating environment. Hence, it is safe to consider ABR a dynamical system. This assumption leads to the application of the Dynamical System Method in both modelling the interactions of the various components of the system (ABR) and describing the behaviour of the scheme as a whole.

Naturally, before modelling any system, there is a need for a clear articulation of how the system operates. In order to achieve this, Chapter 2 develops what we call *Classic Framework*, which describes the way and manner a typical state-of-the-art Adaptive Bitrate Selection module works. After that, we take an incremental approach to improving the classic framework. At the beginning of each chapter, between Chapter 3 to Chapter 5, we present a new framework that captures our understanding of the system at that stage.

Fundamentally, the behaviour of a dynamical system is described by the state vector. A state vector is a set of parameters that collectively define exactly the state of a system. One import assumption of a dynamical system method is that $\mathcal{S}_{i+1} = f(\mathcal{S}_i)$. In other words, the current state uniquely determines all future states. To adequately describe an ABR system, this thesis argues that this particular aspect of dynamical system method is a vital requirement. Because as we have continuously emphasised throughout the thesis, the primary purpose of video quality adaptation is improving the user experience. And user experience is not an isolated event, the current state of a user's perception of the quality of the delivered video depends on the previous actions taken by an ABR. Furthermore, implicit in this assumption is the presence of an evolution rule that correctly specifies the future states that follow from the current state. Therefore, the first goal we set in Chapter 1 is not to only carefully articulate the valid system states, which is a critical task, but also to define the pattern of state transition rule that maximises user experience.

However, before determining the state vector and the evolution rule, the chapter (Chapter 3) reformulates the classic framework upon which most of the existing state-of-the-art ABR modules are based. The proposed framework, through the Policy 1, breaks the cyclic relationship between throughput estimation module and chunk request scheduling function. The Policy 1 is aimed at ensuring that regardless of the ABR algorithm used in video rate selection, TCP is allowed to reach a steady state without any interference from the scheduling function.

Based on the derived Framework 1, the state vector ($\mathcal{S}$) is defined as a set of two parameters: the buffer level and the throughput. While it is relatively easy to measure and control buffer accurately this is not the case with the available bandwidth. So, to simplify the task of modelling a weaker definition of the system state is proposed in Definition 3.3, in which buffer state change becomes the sole indicator of state change. However, for this definition to be valid Policy 2 must be satisfied, that is, throughout the streaming session an ABR module must not select a video rate that is greater than the available throughput. Building on top of the Definition 3.3, a rate map that determines the state transition path is developed in Equation 3.11. The rate map takes an ABR from the lowest to the highest available video rate as the buffer (state) changes. Simply put, the function (Equation 3.11) takes the current state $\mathcal{S}_i$

(buffer level) and returns the video rate to be downloaded, in the process resulting in a new state $\mathcal{S}_{i+1}$ that can be the same or different from the previous state $\mathcal{S}_i$. After receiving ith chunk, an ABR evaluates Equation 3.11, using the current buffer level, to obtain the video rate of chunk $i+1$. It is worth emphasising that Equation 3.11 is algorithm independent, this guarantees that regardless of the algorithm used, which usually depends on the QoE metric that and ABR designer may want to optimise, the video rate evaluation remains the same.

Equation 3.11 only tells us how to act, given a change in the system state, the actual video rate change depends on the rate evolution constant α. Which as seen in Equation 3.13, in turn, depends on three factors: (1) the maximum video rate that a client can display, (2) the size of the allocated buffer space and (3) the current buffer level. At this point, we can conclude that we have identified the valid system states and the pattern of transition between the defined states, or simply put objective I has been achieved.

At this point it is reasonable to ask, to what extent does achieving the first objective helps in validating the hypothesis? To answer this question, Chapter 3 takes a two-pronged approach: analytical and experimental. It turns out that to validate the hypothesis, all that has to be done is to show that the QoE metrics presented at Section 2.8 are improved by adopting the proposed scheme.

Using the analytical method in Section 3.4, we show that provided the available network capacity is equal to or greater than the maximum video rate; the proposed rate map will converge at the highest video rate. Furthermore, its is shown that the system is stable at the convergence point. In other words, the model guarantees that a client will be streaming, after the convergence time, at maximum video rate without suffering for video rate fluctuation. To verify this proof experimentally, two players: a buffer-based [14] and a throughput-based [22] are modified to work with the proposed rate map. Various experiments are conducted within both wired and wireless environments. The results show that adopting the proposed model increases the average video rate, capacity utilisation and the stability of a streaming session. At the same time, it reduces both the start-up delay and the convergence time. All these happen without any adverse impact on the player's fairness both to other players and background traffic. Evidently, using the model has an upward effect on QoE. Hence, the model has contributed in validating our hypothesis.

6.2.2 Objective II

In Chapter 3, we made an assumption, in deriving the video evolution rule, that the available capacity is sufficient to cover the highest video rate, and left it to the algorithms built on top of the rate map to take care of the contrary situation. While this simplifies the modelling process, it has some drawbacks, such as:

1. The model does not consider the prevailing QoE metrics when evolving the video rate.
2. The model has a large buffer requirement.
3. The model lacks a mechanism for handling fluctuation in system capacity.

To solve the first problem highlighted above, in Chapter 4, we introduce the evolving state of user experience as a requirement for making any video rate selection decision. In other words, an adaptation module must take input from the user experience subsystem in addition to the system-level metrics when making any video rate selection decision. With this policy, we have two planes of reference: the system and the user experience. To ease the task of modelling, we treat each of the planes as an autonomous subsystem. Furthermore, to have an algorithm agnostic formulation we develop two state equations with each describing the general behaviour of the system at a particular plane. Building on top of these state equations, a new framework called Unified Framework is derived.

The unified framework is built on top of the Definition 3.1. The framework starts by making the buffer occupancy the main factor and the estimated throughput the adjustment factor. Using this design principle, we derive the two rate maps in Equation 4.11 and 4.16, with the former describing the video rate evolution map in an environment where the network capacity is both stable and abundant and the latter describing the same for a resource-constrained context. In these rate maps, an adaptive rate evolution constant is used, which results in a significant reduction in buffer requirement without any adverse impact on the user experience. Considering this, we can conclude that the second problem of the model proposed in Chapter 3 is solved. To solve the third issue, the impact of throughput fluctuation on the video rate evolution is derived in Equation 4.16. This model helps an ABR algorithm to make a decision not only base on the amount of the available resource but its stability also.

To model the dynamics of the user experience, a QoE definition is used, in which user experience is defined as the difference between the delight and the annoyance suffered by a user during a streaming session. In the modelling process, several QoE metrics are considered in defining the two pivotal words in the definition (delight and annoyance). While it is easy to describe the delight experienced by a user which is represented by the degree of the perceptual video quality, this is not the case with annoyance. Video stalls, video quality fluctuations and start-up delay are the most prominent metrics used in measuring a user's dissatisfaction (for more detail see Section 2.8). Since video rebuffering is the most annoying of all the known factors affecting QoE [15], we insist, through Policy 3, that any video rate map that results from the improved model must not unnecessarily rebuffer. That is provided the available network capacity is equal or greater than the minimum video rate the proposed rate map should not result in video freeze. With this policy guaranteed, any rebuffering event that occurs is beyond the control of a typical ABR algorithm, which makes such information unactionable. Thus, we do not include rebuffering in the QoE model. A further consideration is the start-up delay. First, as we demonstrated in Section 2.8 there is no consensus as to the actual impact of start-up delay on the overall user experience. Secondly, by guaranteeing Policy 3 it is most likely going to be a one-time event. Therefore, start-up delay is not included in the modelling process. Nonetheless, it is worth noting that by reducing the buffer requirement we have reduced it [1]. Consequently, a model of the impact of video quality fluctuation on user experience that takes into account the frequency, amplitude, and direction of the video quality switch, as well as the user forgiveness, is derived.

Furthermore, using the analytical approach, we show that when the network capacity is both stable and abundant, the model will always converge at the highest video rate. Hence, guaranteeing the user the maximum video quality. And if the available network capacity can not sufficiently cover the maximum video rate, the model still converges at the video rate that equals the average throughput. Additionally, we prove that provided the available network resource is more than the minimum video rate the derived rate maps will not rebuffer. Finally, when the model converges at the maximum video rate, the convergence is shown to be asymptotically stable. However, if system converges at a video rate that is less than the maximum, the system is shown to be locally stable. In other words, there is a likelihood of video rate fluctuation. Therefore, to help an algorithm designer handle this situation, the impact of such variation is also modelled. In summary, in Chapter 4 the dynamics of the individual components of an ABR system is modelled, and rules that govern their interactions are formulated. Therefore we can conclude that objective II has been achieved.

Through the presented analysis, it is evident that the proposed design helps in eliminating unnecessary video stalls, reducing start-up delay, and ensuring that video rate fluctuation happens only when it is unavoidable. Hence, the realisation of the objective II contributes to the validation of the hypothesis.

6.2.3 Objective III

The last task we set is to demonstrate how the models developed can be used to design and develop effective HTTP Adaptive Streaming service. In Chapter 3, two start-of-the-art ABR schemes are modified to work with the proposed rate map. Furthermore, in Chapter 5, a novel cooperative streaming service is designed and implemented. The first part of the cooperative algorithm gathers the required information needed

[1]Recall from Chapter 2 there is a proportional relation between buffer size and start-up delay

to make a video rate selection decision. The second part makes the decisions. In summary, all the participating clients, throughout a streaming session, are required to periodically send their network state and location information to the bandwidth lookup server. When a player intends to use the service, through the first part of the algorithm, requests information about the network state along its current route. Using the received data, and the client's measured user experience and system-level parameters, the second part of the algorithm selects a video rate that is least likely to cause future degradation in user experience. Therefore, from this discussion, we can conclude that objective III has been achieved.

All the proposed algorithms are experimentally evaluated. In summary, the various presented results show an improvement in both the magnitude and the stability of the video quality, even when requesting a lower video rate. Furthermore, the schemes help eliminate unnecessary rebuffering, cut down the volume of data download and reduce the start-up delay. Hence, the developed models and the implemented algorithms increase the overall user satisfaction. Therefore, we can finally conclude our hypothesis is validated.

6.3 Implementation

To evaluate the proposed algorithms several HAS players have been studied, but none of them is found to fit our requirements completely, which are the availability of a customisable statistical information logging module, the mode for disabling video rendering module, the support for multiple algorithms and the capability of processing geolocation information. Furthermore, we found that while the behaviour of most of the open source players, such as DASHJS, keep on changing because of the rapid phase of innovation in this field, the stable players tend to have proprietary components that can not be altered. Therefore, an MPEG-DASH compliant streaming service is developed.

As part of the service, a player is designed and built. The player is written in Python and uses Request library to handle HTTP transactions. To ensure that the same implementation of the player is used throughout our evaluation, we use a modular architecture, which allows us to implement and test each module independently. Additionally, the modular design makes it easy to change video rate adaptation algorithm at any stage of our experimentation. The modules implemented are communication module that handles all communications between the player and the server, statistical information logging module, adaptation module, buffer tracker and throughput estimation module. Whenever the player is used in cooperative mode additional modules are activated, e.g. geolocation processing module, and mobility simulator.

In addition to the player, an automation suite is developed, which is implemented using the combination of Python, Octave and BASH. The suite enables us to experiment in a reproducible way. Furthermore, with it, we are able to perform hundreds of experiments, process the results and plot the relevant graphs in one go without human intervention. The suite uses Dummynet for network emulation, tcpdump to capture traffic for offline analysis and lsof to map applications to port numbers. Moreover, for result processing and presentation, several applications are built and integrated into the suite, such as throughput calculator that takes the network trace file and return the corresponding TCP throughput, a statistical package that is used to compute various statistical reports and visualisation package that plots the final result. The entire developed service has been released as open source for other researchers to use and modify freely [2].

6.4 Integration

A further crucial aspect is how the proposed adaptation algorithms can be integrated into a production level open source player. There are some open source DASH players available, e.g. DASH plugin for VLC [166], ExoPlayer developed by YouTube for Android [167] and Dash.js the official reference player of the DASH Industry Forum [168]. While both VLC and Dash.js are completely open source and are governed by a very

[2]https://github.com/yusufsani/Bio-inspired-Player

liberal license, the VLC plugin only works with the Firefox browser. Therefore, we only discuss how proposed algorithms can be integrated into Dash.js.

Dash.js is a JavaScript-based player. To change the adaptation algorithm to the ones proposed in Chapter 3. The first thing to do is to modify the *Bandwidthestimator* class, which is the class that contains the throughput estimation logic. Its default implementation derives the estimated throughput from the moving average of the per-chunk throughput of n-number of chunks. The class needs to implement Equation 3.2. For any developer willing to change the video rate selection algorithm the Dash.Js provides an interface called *BaseAdaptationLogicis* that must be implemented. The interface provides only one method called *switchRepresentation()*. Therefore, a new class that inherits the *BaseAdaptationLogicis* and overrides this method with the implementation of any of the proposed algorithm is required. For the buffer-based player, a developer should add a new method here that implements the rate map. Please note that buffer state reading are obtained by listening to the *bufferFillStateListener()*.

For the cooperative player, in addition to the above change, a developer should modify the *segmentRequester* class, which is responsible for managing the communication between the player and the server. The new implementation must now include the logic that will handle request-response transactions between the player and the Bandwidth Lookup server. Next, we suggest that a developer should create a new class that manages the processing of the received geolocation data. Furthermore, for the robustness factor, either a method is added to the *Bandwidthestimator* class, or a new class is created with such method. To maintain the player's modular design we recommend the latter option.

6.5 Revisiting the Related Work

Earlier studies, as extensively discussed in Chapter 2, took a variety of approaches to enhance the performance of HTTP Adaptive Streaming service. Some of these have a complementary relationship with the schemes proposed in this thesis, while other are orthogonal to it.

To the best of our knowledge, the work presented in this thesis is first to employ both the dynamical system method and a bio-inspired approach to model and design a video rate selection problem. Though the work presented in [14, 23] did use stability analysis (which is an integral part of the dynamical system method) the proof was based on induction. In the following, we discuss how this work presented in this thesis relates to the resource based, central control plane based, Quality of Experience-based research.

6.5.1 Resource-based ABR

Resource-based approaches, such as throughput-based [13, 19, 18, 62, 87] and buffer-based [14, 23] ABR schemes are the earliest proposed video rate selection algorithms. Throughput-based algorithms, as discussed in Section 2.3, primarily rely on the assumption that all that is needed to build an efficient ABR algorithm is the capability to monitor an accurately estimate the available bandwidth [20, 26]. A key feature of this thesis is that we admit the importance of an accurate estimation of throughput in choosing the right video rate to request. Furthermore, we consider most of the techniques proposed in the literature, for instance [19, 21, 75] to be complementary to our work. However, this thesis argues that an accurate estimation of throughput alone is not a sufficient condition for guaranteeing a high-level user experience.

Conversely, buffer-based ABR algorithms use buffer state changes in making video rate selection decisions without taking into consideration the available bandwidth. While these algorithms can provide superior performance with regard to some QoE metrics (see Chapter 2 for detail), they usually have a significant buffer requirement and lack a formal definition of the relationship between the available video rates and the buffer occupancy, which is necessary for their operation. To help in this regard, the thesis proposed several video rate maps and incorporated them into the most prominent buffer-based ABR schemes e.g., [14, 23]. Furthermore, the thesis shows that buffer occupancy alone is not a sufficient factor for ensuring high QoE, especially when the transmission channel is unreliable. The thesis concludes that for an effective ABR algorithm both buffer occupancy and throughput in addition to QoE-related information are needed.

Mobile devices, such as smartphones and tablets, are usually energy constrained and video streaming is a power intensive exercise. Several researchers have found a significant increase in power expenditure when streaming video [169, 170, 171]. In fact, video streaming can consume as much as twice the energy of playing the same content off-line [169]. Understandably, this shortens the battery life and consequently increases the probability of the battery running low in the middle of a streaming session [169]. To prolong the battery life some researchers [70, 72, 172, 173] have attempted to use power consumption as a factor for video rate selection. The authors of [70] optimise power consumption by minimising the number of active periods while streaming over 3G/4G. They achieve this through the dynamic management of the buffer while relying on the user's view history and network state. In [72], a bundled chunk download strategy is presented. Put simply, a client downloads a set of chunks (as a bundle) and then waits until its buffer is depleted to a certain threshold before requesting another bundle. The authors [174] propose a similar approach where a client pre-fetches a significant amount of video content then enters an OFF period until the buffer shrinks to a predefined threshold. We consider these approaches to be complementary efforts, for instance, an appropriate model of power depletion rate can be easily included into Equation 4.13 as an additional exogenous variable.

6.5.2 Network and Server-assisted Approach

A client-side ABR design distributes the task of both resource monitoring and the video rate selection. This design approach evidently enhances scalability. However, it results in an *opportunistic* ABR algorithm. Put differently, a decentralised design of the ABR results in an algorithm that only optimises its performance [95]. Furthermore, with a client-driven architecture, it is hard for a service provider to ensure a consistent and guaranteed Quality of Experience. To solve this problem, there has been a growing attempt on the network and server-assisted solutions.

Server and Network Assisted DASH (SAND) [175] is an active effort of the Moving

Picture Experts Group (MPEG) to standardise the network-to-client and network-to-network exchange of quality-related information with the aim to assist DASH clients in making video rate selection decision. Also, in [67] a central control plane that aggregates measurements from many clients is proposed. The scheme ensures that performance across clients is globally optimised. Furthermore, in [1] a control plane is used to orchestrate the monitoring and the measurement process of video streams in a network. The goal here is to ensure network-wide QoE fairness is achieved. In a similar vein, the authors of [68] propose a network control plane with the aim of maximising network-wide QoE and bandwidth utilisation. They rely on bandwidth reservation on a per-flow basis to achieve this. Also, [69] propose an SDN-based architecture that allocates bandwidth to competing clients based on both content complexity of the requested video and the buffer status of the individual clients.

Several researchers have deployed the HAS adaptation logic at the server [176, 177, 178]. The authors of [99, 98] have used the HTTP 2.0 server push capabilities to send video chunks to a client pre-emptively. In this scheme, a server either pushes a k chunks after the receipt of a request from a client or continuously sends chunks back-to-back until requested otherwise by a client. However, in [87] a server-side traffic shaping technique is used to aid client better estimated its fair share of the available bandwidth by neutralising the OFF period. We consider the work proposed in this thesis to be orthogonal to both server and network assisted approaches.

6.6 Shortcomings and Future Work

In this thesis, a set of frameworks and models are presented with the aim of assisting the designers of future video content delivery platforms. Furthermore, several prototype video delivery services have been implemented to demonstrate how the models can be used in building innovative services or enhancing the existing schemes. While thorough evaluation has confirmed the benefit of the developed schemes, more work can improve their effectiveness.

As discussed at the beginning of this thesis, the focus of the work is efficient delivery of VoD using HTTP-based adaptive video streaming technology. This obviously restricts our attention to the mechanisms that are relevant to VoD architects. Live streaming has more stringent timing requirements than VoD services. Since buffer plays a crucial role in all of the discussed ABR schemes the delay introduced through this would be probably too high to meet the timing requirements of live streaming. Furthermore, by relying on a distributed approach to video rate selection, using our design a service provider, typically an ISP or a Cable TV company, will have virtually no control over resource allocation, and since video traffic is heavyweight, this may not be the best option of any provider willing to provide a managed service.

Concerning the actual modelling, the rate evolution maps derived in the thesis are all reactive, that is, they rely on the current state of the buffer without taking into consideration the potential for future development. This reliance on historical data is what necessitates the large buffer requirement. Furthermore, throughout the modelling process, we have implicitly assumed that we can handle all the stochastic processes, for example, throughput fluctuation, using deterministic approach. One possible extension of this work is to gather enough data, so that we can use probabilistic methods to, for example, model the impact of throughput variation on the stability of the system.

For the bio-inspired approach, the impact of competition amongst the streaming the players is not included. Since competition for the available resource can be a serious source of fluctuation in video quality. The bio-inspired design can be improved to cover this design space. Biologists have long investigated how competition amongst species affect their survival rate, the most well-known of these efforts is the Lotka-Volterra model [179]. This model can be used to extend the proposed bio-inspired approach. However, such an approach may require a central control plane for gathering global information like the total number of participating client and available resource.

In the evaluation of the cooperative algorithm, for the trace used we had to artificially introduce some noise to make the experimentation more realistic. In future, we intend to collect more realistic data sets, for example through crowd-sourcing. This will have a couple of benefits, for example, we can better model the distribution of the trace, as such provide more advanced lookup service.

Chapter 7

Conclusion

This thesis addresses the fundamental issues involved in Internet based video streaming. Despite the fact that video streaming makes up a substantial part of Internet traffic [5] problems related to efficient delivery have not been satisfactorily solved. Video delivery services have continued to evolve in order to meet the need of content providers that have to cater for different users and their interests, serve a variety of contents from a multitude of sources, and accommodate resource constrained devices using different network technologies. However, the overarching goal has remained constant: to provide the best possible experience to the users in the most cost effective manner.

Recent advances in video content delivery allow adaptable video playback, such that the quality of a video delivered to a user is adjusted to fit its context. The most recent adaptive video delivery scheme that is increasingly being adopted as the de facto standard for Internet-based video streaming is HTTP-based adaptive streaming (HAS). Despite the success of HAS there are still open issues, e.g., when streaming adaptive media, user clients continuously monitor system and network statistics and without feedback from the user experience sub-system decide whether to switch the representation for the next chunk to download, which leads to sub-optimal overall user experience

This thesis takes a holistic analysis of ABR systems. To better manage the complexity of the ABR module, the thesis functionally decomposes the module into a number of sub-components. Then we argue that in order to understand the video quality adaptation problem in HAS a video rate selection algorithm designer needs to pay attention to all the relevant sub-components of the ABR (resource estimation function, chunk request scheduling function, and adaptation module). For this to happen, there is a need for a clear articulation of the relationships and the interactions between these various components.

Throughout the previous chapters, the relationship between these components is formalised. This begins with the definition of the valid system state space, from which the model of the state transition is developed. Iteratively, the models are improved such that at each stage of the iteration previous simplifying assumptions are relaxed. The resulting final model captures all relevant aspects and is representing a realistic video quality dynamics.

Furthermore, after each refinement of the proposed models, some algorithms are derived from the models either by improving the existing state-of-the art or developing novel solutions. The results of evaluations show that incorporating a systematic model that captures the relationship between the components of an ABR can improve the QoE performance of a HAS player.

This Chapter concludes the thesis. First, an overview of the previous chapters is presented. Then the main contributions of the thesis are discussed.

7.1 Overview

The thesis is organised into six different chapters. Chapter 1 begins by introducing adaptive video streaming before presenting a detailed discussion of the challenges facing the state-of-the-art services. Some research goals are then developed. The chapter is then rounded up with a discussion on the scope and the limitation of the study.

Chapter 2 provides an in depth discussion of the background and the related work. It starts with a brief historical account of Internet-based video streaming services. This covers both TCP and UDP based schemes. An overview of HTTP-based streaming services is then presented. This is followed by a detailed discussion on different techniques of resource monitoring and measurement, scheduling chunk requests, and video quality adaptation function. After these, the chapter discusses context management and the QoE factors that are specifically relevant to adaptive streaming services.

Chapter 3 gives the motivation on why a careful selection of system state space is important. Then the valid states are defined and explained. Since video rate selection decisions are not made in isolation, a model of the video rate evolution trajectory that covers all stages of the streaming session is developed. From this, a number of state-of-the-art ABR algorithms are modified to work with the new model. The chapter concludes with a detailed discussion of the performance evaluation results.

This is followed by Chapter 4, which improves the framework presented in the previous chapter. Using system dynamics methodology a generalised model of the dynamics of adaptive video streaming is presented, from which a new framework that avail adaptation module with both the system-level information and QoE metrics is developed. After that, the proposed model of the video quality dynamics in both resource-abundant and constrained context is discussed.

In Chapter 5, the thesis presents a cooperative algorithm built on top the framework developed in the previous chapter. A prototype is then implemented. This is followed by a detailed evaluation of the proposed service.

Chapter 6 brought together and reflected on the technical contributions of previous chapters. The chapter also discussed how the work presented in the thesis can be adopted and deployed in practice. It then concluded with a presentation on how the models and frameworks developed in the thesis compare with other relevant research.

7.2 Major Contributions

Adaptive Bitrate Selection Frameworks

The thesis proposes a series of frameworks that provide guidelines to be used in the design and implementation of the next generation adaptive video streaming services. The frameworks are derived through a detailed investigation of the strengths and shortcomings of the state-of-the-art video delivery services and user requirements.

Models of Video Rate Evolution

Video rate transition models have been developed to ensure that regardless of the type of technique used in realising the video rate adaptation the pattern of video quality change remains the same. The models benefit from the patterns of video quality change that are known to have an impact on the user experience. Furthermore, the behaviour of the developed models has been thoroughly investigated, both analytically and through experiments. The models are shown to guarantee that video rate selection decision will always convergence at the optimal video rate and is stable.

Bio-inspired Adaptive Streaming Technique

The thesis developed a new approach that balances multiple key video quality factors using a bio-inspired optimisation design with the aim of maximising the overall user experience of a video playback session. The design is based on an analogy, which considers the video quality level to be a species whose growth an adaptive streaming algorithm designer is interested in, and the playback buffer to be its habitat. The approach was found to be effective in reducing video rate instability and increasing the average video rate. Apart from introducing a new approach to the design of video rate adaptation, it is easily extensible to the scenario whereby a central control is used to orchestrate fairness.

Model of Video Quality Fluctuation

A novel model of the impact of video quality fluctuation on user experience that takes into account the frequency, amplitude, and direction of the video quality switch, as well as the user forgiveness, is developed. The model can be used to provide an additional parameter to the adaptation model, such that when making a video rate selection decision not only the system-level parameters are considered but also the evolving state of the user experience. Further, the model can be used as an independent metric to measure the level of annoyance a user suffers as the result of video quality fluctuation.

Cooperative Streaming Service

A novel cooperative adaptive video streaming service, which ensures that all participating clients contribute towards the enhancement of the effectiveness of the delivered service is developed. The service demonstrates how to put the various models proposed in the thesis to build a scheme that ensures that any decision an adaptive player makes will enhance the video quality perceived by a user, and will not result in a future degradation in the user experience.

7.3 Other Contributions

Improvement of Existing Services

A number state-of-the-art adaptive video streaming algorithms have been improved using the proposed models. The algorithms are found to benefit from the improvement in terms of a number of QoE metrics. While this serves to demonstrate how the models can be used in practical systems, it also provides the HAS research community with fully functional services.

Testbeds System

Many prototype players have been developed purposely to evaluate the effectiveness of the proposed models and algorithms. Furthermore, some automation tools have been built that use the prototype player and the embedded statistical analysis tools to allow for easy replication of all the evaluations done in the thesis. All code has been released as open source.

Bibliography

[1] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide qoe fairness using Openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, ser. FhMN '13. ACM, 2013, pp. 15–20.

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

[3] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4. ACM, 1994, pp. 58–67.

[4] "Sandvine global internet phenomena report," White Paper, Sandvine, Jun. 2013. [Online]. Available: http://www.sandvine.com/downloads/documents/Phenomena_1H_2013 /Sandvine_Global_Internet_Phenomena_Report_1H_2013.pdf

[5] "Cisco visual networking index: Forecast and methodology,2014-2019," White Paper, Cisco, April 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[6] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard, "Optimized adaptive streaming representations based on system dynamics," *arXiv*, 2014.

[7] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems.* ACM, 2011, pp. 133–144.

[8] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a.* IEEE, 2011, pp. 1–8.

[9] R. Pantos, "HTTP live streaming draft-pantos-http-live-streaming-11," Internet Draft, IETF, April 2012. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-11

[10] "3rd generation partnership project; technical specification group services and system aspects; transparent end-to-end packet-switched streaming service (PSS); protocols and codecs (release 12) 3GPP TS 26.234 v12.0.0," Tech. Rep., march 2013.

[11] S. Akhshabi and C. Dovrolis, "The evolution of layered protocol stacks leads to an hourglass-shaped architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 206–217.

[12] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *Consumer Electronics, IEEE*, vol. 58, no. 1, pp. 78–85, 2012.

[13] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video.* ACM, 2012, pp. 9–14.

[14] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming," in *Proceedings*

of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking. ACM, 2013, pp. 9–14.

[15] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 1, pp. 469–492, 2015.

[16] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, 2012.

[17] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in http adaptive streaming: Friend or foe?" in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop.* ACM, 2014, p. 31.

[18] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *Proceedings of the second annual ACM conference on Multimedia systems.* ACM, 2011, pp. 169–174.

[19] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies.* ACM, 2012, pp. 97–108.

[20] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference.* ACM, 2012, pp. 225–238.

[21] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies.* ACM, 2012, pp. 109–120.

[22] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Packet Video Workshop (PV), 2012 19th International*. IEEE, 2012, pp. 173–178.

[23] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. ACM, 2014, pp. 187–198.

[24] K. Miller, D. Bethanabhotla, G. Caire, and A. Wolisz, "A control-theoretic approach to adaptive video streaming in dense wireless networks," *Multimedia, IEEE Transactions on*, vol. 17, no. 8, pp. 1309–1322, Aug 2015.

[25] T.-Y. Huang, "A buffer-based approach to video rate adaptation," Ph.D. dissertation, Stanford University, 2014.

[26] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 157–168.

[27] T. Kupka, P. Halvorsen, and C. Griwodz, "Performance of on-off traffic stemming from live adaptive segmented HTTP video streaming." in *LCN*. Citeseer, 2012, pp. 401–409.

[28] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637–647, 2006.

[29] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "QDASH: a QoE-aware DASH system," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 11–22.

[30] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," *Circuits*

and Systems for Video Technology, IEEE Transactions on, vol. 11, no. 3, pp. 269–281, 2001.

[31] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 1998.

[32] L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (rsvp)–version 1 functional specification," *Resource*, 1997.

[33] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "A transport protocol for real-time applications," RFC 3550, IETF, July 2003.

[34] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," RFC 2326, 1998.

[35] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 54–63, 2011.

[36] N. Bouten, M. Claeys, S. Latre, J. Famaey, W. V. Leekwijck, and F. D. Turck, "Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–7.

[37] M. Ellis, "Understanding the performance of internet video over residential networks," Ph.D. dissertation, University of Glasgow, 2012.

[38] Z. Yetgin and G. Seckin, "Progressive download for 3g wireless multicasting," in *Future Generation Communication and Networking FGCN 2007*, vol. 1. IEEE, 2007, pp. 289–295.

[39] N. G. Feamster, "Adaptive delivery of real-time streaming video," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[40] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for internet video streaming," *RFC 2733*, 1999.

[41] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Real-time system for adaptive video streaming based on svc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1227–1237, 2007.

[42] J.-C. Bolot and T. Turletti, "A rate control mechanism for packet video in the internet," in *INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE.* IEEE, 1994, pp. 1216–1223.

[43] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[44] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *IEEE transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1204–1217, 2007.

[45] Y. Pei and J. W. Modestino, "Multi-layered video transmission over wireless channels using an adaptive modulation and coding scheme," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 2. IEEE, 2001, pp. 1009–1012.

[46] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an experimental investigation of QUIC," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing.* ACM, 2015, pp. 609–614.

[47] L. Popa, A. Ghodsi, and I. Stoica, "HTTP as the narrow waist of the future internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks.* ACM, 2010, p. 6.

[48] J. Postel, "RFC 793: Transmission control protocol, september 1981," *Standard*, vol. 8.

[49] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[50] T. Kim, N. Avadhanam, and S. Subramanian, "Dimensioning receiver buffer requirement for unidirectional vbr video streaming overTCP," in *Image Processing, 2006 IEEE International Conference on.* IEEE, 2006, pp. 3061–3064.

[51] K. J. Ma, R. Bartoš, and S. Bhatia, "A survey of schemes for internet-based video delivery," *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1572–1586, 2011.

[52] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via TCP: an analytic performance study," in *Proceedings of the 12th annual ACM international conference on Multimedia.* ACM, 2004, pp. 908–915.

[53] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 6, pp. 841–851, 2004.

[54] B. W. Wah, X. Su, and D. Lin, "A survey of error-concealment schemes for real-time audio and video transmissions over the internet," in *Multimedia Software Engineering, 2000. Proceedings. International Symposium on.* IEEE, 2000, pp. 17–24.

[55] T. Hoßfeld, R. Schatz, and U. R. Krieger, "QoE of YouTube video streaming for current internet transport protocols," in *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance.* Springer, 2014, pp. 136–150.

[56] T. Ho"ß"feld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia, "Identifying qoe optimal adaptation of HTTP adaptive streaming based on subjective studies," *Computer Networks*, vol. 81, pp. 320–332, 2015.

[57] D. Robinson, "Live streaming ecosystems," *Advanced Content Delivery, Streaming, and Cloud Services*, pp. 33–49, 2014.

[58] "Open iptv forum release 2 specification volume 2a - http adaptive streaming," White Paper, Open IPTV Forum, April 2016. [Online]. Available: http://www.oipf.tv/web-spec/volume2a.html

[59] "MPEG DASH specification (ISO/IEC 23009-1:2012) dynamic adaptive streaming over HTTP (DASH) part 1: Media presentation description and segment formats," march 2012.

[60] A. Zambelli, "IIS smooth streaming technical overview," Technical Report, Microsoft Corporation, 2009.

[61] Adobe, "OSMF player," June 2013. [Online]. Available: http://www.osmf.org

[62] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies.* ACM, 2011, p. 25.

[63] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of YouTube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012.

[64] "SLExtensions adaptive streaming," March 2014. [Online]. Available: https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/ AdaptiveStreaming

[65] Y. Sani, A. Mauthe, and C. Edwards, "Modelling video rate evolution in adaptive bitrate selection," in *The IEEE International Symposium on Multimedia (ISM 2015)*, Dec. 2015.

[66] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.

[67] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, “A case for a coordinated internet video control plane,” in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication.* ACM, 2012, pp. 359–370.

[68] G. Cofano, L. De Cicco, and S. Mascolo, “A control architecture for massive adaptive video streaming delivery,” in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming.* ACM, 2014, pp. 7–12.

[69] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla, “SDN based QoE optimization for HTTP-based adaptive video streaming,” in *2015 IEEE International Symposium on Multimedia (ISM)*, Dec. 2015, pp. 120–123.

[70] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, “GreenTube: Power optimization for mobile videostreaming via dynamic cache management,” in *Proceedings of the 20th ACM International Conference on Multimedia*, ser. MM ’12. ACM, 2012, pp. 279–288.

[71] J. Chen, A. Ghosh, J. Magutt, and M. Chiang, “QAVA: quota aware video adaptation,” in *Proceedings of the 8th international conference on Emerging networking experiments and technologies.* ACM, 2012, pp. 121–132.

[72] G. Tian and Y. Liu, “On adaptive HTTP streaming to mobile devices,” in *Packet Video Workshop (PV), 2013 20th International*, Dec. 2013, pp. 1–8.

[73] M. Jain and C. Dovrolis, “Ten fallacies and pitfalls on end-to-end available bandwidth estimation,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement.* ACM, 2004, pp. 272–277.

[74] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, “Bandwidth estimation: metrics, measurement techniques, and tools,” *Network, IEEE*, vol. 17, no. 6, pp. 27–35, 2003.

[75] X. Qiu, H. Liu, D. Li, S. Zhang, D. Ghosal, and B. Mukherjee, "Optimizing http-based adaptive video streaming for wireless access networks," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on.* IEEE, 2010, pp. 838–845.

[76] 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," *arXiv*, 2013.

[77] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1. ACM, 2007, pp. 97–108.

[78] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[79] V. Ramamurthi and O. Oyman, "Link aware http adaptive streaming for enhanced quality of experience," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Dec. 2013, pp. 1675–1680.

[80] V. Ramamurthi, O. Oyman, and J. Foerster, "Using link awareness for HTTP adaptive streaming over changing wireless conditions," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*, Feb 2015, pp. 727–731.

[81] T. Nguyen and S.-C. S. Cheung, "Multimedia streaming using multiple TCP connections," in *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International.* IEEE, 2005, pp. 215–223.

[82] R. Kuschnig, I. Kofler, and H. Hellwagner, "Improving internet video streaming performance by parallel TCP-based request-response streams," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE.* IEEE, 2010, pp. 1–5.

[83] S. Gouache, G. Bichot, A. Bsila, and C. Howson, "Distributed & adaptive http streaming," in *Multimedia and Expo (ICME), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–6.

[84] M. H. Alizai, O. Landsiedel, J. A. B. Link, S. G"̈o"tz, and t. . B. T. o. B. L. Wehrle, Klaus", in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. ACM, 2009, pp. 71–84.

[85] B. J. Villa and P. E. Heegaard, "Group based traffic shaping for adaptive HTTP video streaming by segment duration control," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, March 2013, pp. 830–837.

[86] C. Aurrecoechea, A. T. Campbell, and L. Hauw, "A survey of qos architectures," *Multimedia systems*, vol. 6, no. 3, pp. 138–151, 1998.

[87] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video.* ACM, 2013, pp. 19–24.

[88] R. Houdaille and S. Gouache, "Shaping http adaptive streams for a better user experience," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. ACM, 2012, pp. 1–9.

[89] J. G. Apostolopoulos, W.-t. Tan, and S. J. Wee, "Video streaming: Concepts, algorithms, and systems," 2002.

[90] A. Dua and N. Bambos, "Buffer management for wireless media streaming," in *Global Telecommunications Conference*, ser. GLOBECOM'07. IEEE, 2007, pp. 5226–5230.

[91] Y. Xu, Y. Zhou, and D.-M. Chiu, "Analytical QoE models for bit-rate switching in dynamic adaptive streaming systems," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 12, pp. 2734–2748, Dec. 2014.

[92] A. El Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "QoE-based traffic and resource management for adaptive HTTP video delivery in LTE," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 25, no. 6, pp. 988–1001, June 2015.

[93] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, 2012.

[94] D. Johansen, H. Johansen, T. Aarflot, J. Hurley, Å. Kvalnes, C. Gurrin, S. Zav, B. Olstad, E. Aaberg, T. Endestad *et al.*, "Davvi: A prototype for the next generation multimedia entertainment platform," in *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 2009, pp. 989–990.

[95] T.-Y. Huang, "A buffer-based approach to video rate adaptation," Ph.D. dissertation, Stanford University, 2014.

[96] N. Gautam, H. Petander, and J. Noel, "A comparison of the cost and energy efficiency of prefetching and streaming of mobile video," in *Proceedings of the 5th Workshop on Mobile Video*, ser. MoVid '13. ACM, 2013, pp. 7–12.

[97] B. J. Villa and E. Heegaard, Poul, "Improving perceived fairness and QoE for adaptive video streams," in *ICNS 2012, The Eighth International Conference on Networking and Services*, 2012, pp. 149–158.

[98] R. Huysegems, T. Bostoen, P. Rondao Alface, J. van der Hooft, S. Petrangeli, T. Wauters, and F. De Turck, "HTTP/2-based methods to improve the live experience of adaptive streaming," in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, ser. MM '15. ACM, 2015, pp. 541–550.

[99] S. Wei and V. Swaminathan, "Low latency live video streaming over HTTP 2.0," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. ACM, 2014, pp. 37:37–37:42.

[100] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori, “DASH fast start using HTTP/2,” in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV ’15. ACM, 2015, pp. 25–30.

[101] S. Wei and V. Swaminathan, “Cost effective video streaming using server push over HTTP 2.0,” in *Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on*, 2014, pp. 1–5.

[102] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. F. Hansen, and j. P. I. C. Engelstad, Paal”, “Using bandwidth aggregation to improve the performance of quality-adaptive streaming,” vol. 27, no. 4, pp. 312–328, 2012.

[103] D. Kaspar, K. Evensen, P. Engelstad, A. F. Hansen, P. Halvorsen, and C. Griwodz, “Enhancing video-on-demand playout over multiple heterogeneous access networks,” in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE.* IEEE, 2010, pp. 1–5.

[104] D. Kaspar, K. Evensen, P. Engelstad, and A. F. Hansen, “Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces,” in *Communications (ICC), 2010 IEEE International Conference on.* IEEE, 2010, pp. 1–5.

[105] C. Liu, I. Bouazizi, and M. Gabbouj, “Parallel adaptive HTTP media streaming,” in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on.* IEEE, 2011, pp. 1–6.

[106] L. De Cicco and S. Mascolo, “An experimental investigation of the Akamai adaptive video streaming,” in *HCI in work and learning, life and leisure.* Springer, 2010, pp. 447–464.

[107] T. Abdelzaher, Y. Diao, J. L. Hellerstein, C. Lu, and X. Zhu, “Introduction to control theory and its application to computing systems,” in *Performance Modeling and Engineering.* Springer, 2008, pp. 185–215.

[108] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proceedings of the second annual ACM conference on Multimedia systems.* ACM, 2011, pp. 145–156.

[109] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. ACM, 2015, pp. 325–338.

[110] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, "Buffer-based smooth rate adaptation for dynamic HTTP streaming," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific.* IEEE, 2013, pp. 1–9.

[111] C. Zhou, X. Zhang, and Z. Guo, "A control theory based rate adaption scheme for DASH over multiple servers," in *Visual Communications and Image Processing (VCIP), 2013.* IEEE, 2013, pp. 1–6.

[112] K. Miller, D. Bethanabhotla, G. Caire, and A. Wolisz, "A control-theoretic approach to adaptive video streaming in dense wireless networks," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1309–1322, Aug. 2015.

[113] N. Bouten, J. Famaey, S. Latre, R. Huysegems, B. Vleeschauwer, W. Leekwijck, and F. Turck, "QoE optimization through in-network quality adaptation for HTTP adaptive streaming," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, Oct 2012, pp. 336–342.

[114] N. Bouten, S. Latre, J. Famaey, W. Van Leekwijck, and F. De Turck, "In-network quality optimization for adaptive video streaming services," *Multimedia, IEEE Transactions on*, vol. 16, no. 8, pp. 2281–2293, Dec. 2014.

[115] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of DASH-based video delivery in networks," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 82–90.

[116] P. Xiong, J. Shen, Q. Wang, D. Jayasinghe, J. Li, and C. Pu, "Nbs: a network-bandwidth-aware streaming version switcher for mobile streaming applications under fuzzy logic control," in *Mobile Services (MS), 2012 IEEE First International Conference on*. IEEE, 2012, pp. 48–55.

[117] D. Vergados, A. Michalas, A. Sgora, and D. Vergados, "A control-based algorithm for rate adaptation in MPEG-DASH," in *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, July 2014, pp. 438–442.

[118] A. Sobhani, A. Yassine, and S. Shirmohammadi, "A fuzzy-based rate adaptation controller for DASH," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV'15. ACM, 2015, pp. 31–36.

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

[120] M. Claeys, S. Latr"é", J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a q-learning-based client quality selection algorithm for http adaptive video streaming," in *Proc. Conference on Autonomous Agents and Multiagent Systems*, May 2013.

[121] M. Claeys, S. Latre, J. Famaey, and F. De Turck, "Design and evaluation of a self-learning HTTP adaptive video streaming client," *Communications Letters, IEEE*, vol. 18, no. 4, pp. 716–719, April 2014.

[122] Y.-L. Chien, K. C.-J. Lin, and M.-S. Chen, "Machine learning based rate adaptation with elastic feature selection for HTTP-based streaming," in *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, June 2015, pp. 1–6.

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

[124] S. Basso, A. Servetti, E. Masala, and J. C. De Martin, "Measuring DASH streaming performance from the end users perspective using neubot," in *Proceedings of the 5th ACM Multimedia Systems Conference*, ser. MMSys '14. ACM, 2014, pp. 1–6.

[125] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 131–138.

[126] K. Mitra, A. Zaslavsky, and C. AAhlund, "Context-aware qoe modelling, measurement, and prediction in mobile computing systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, pp. 920–936, 2015.

[127] F. Metzger, E. Liotou, C. Moldovan, and T. Hoßfeld, "TCP video streaming and mobile networks: Not a love story, but better with context," *Computer Networks*, 2016.

[128] T. Hoßfeld, L. Skorin-Kapov, Y. Haddad, P. Pocta, V. A. Siris, A. Zgank, and H. Melvin, "Can context monitoring improve qoe? a case study of video flash crowds in the internet of services," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM).* IEEE, 2015, pp. 1274–1277.

[129] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a qoe model for http adaptive streaming," in *Quality*

of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on. IEEE, 2014, pp. 111–116.

[130] Y. Zhu, I. Heynderickx, and J. A. Redi, "Understanding the role of social context and user factors in video quality of experience," *Computers in Human Behavior*, vol. 49, pp. 412–426, 2015.

[131] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in high-speed mobility using bandwidth maps," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 603–617, April 2012.

[132] J. Hao, R. Zimmermann, and H. Ma, "GTube: Geo-predictive video streaming overHTTP in mobile environments," in *Proceedings of the 5th ACM Multimedia Systems Conference*, ser. MMSys '14. ACM, 2014, pp. 259–270.

[133] 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?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile 15. ACM, 2015, pp. 57–62.

[134] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 3, p. 24, 2012.

[135] D. Han, J. Han, Y. Im, M. Kwak, T. T. Kwon, and Y. Choi, "MASERATI: Mobile adaptive streaming based on environmental and contextual information," in *Proceedings of the 8th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation; Characterization*, ser. WiNTECH '13. ACM, 2013, pp. 33–40.

[136] R. Dubin, A. Dvir, O. Pele, O. Hadar, I. Katz, and O. Mashiach, "Adaptation logic for http dynamic adaptive streaming using geo-predictive crowdsourcing for mobile users," *Multimedia Systems*, pp. 1–13, 2016.

[137] B. Taani and R. Zimmermann, "Spatio-temporal analysis of bandwidth maps for Geo-predictive video streaming in mobile environments," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 888–897.

[138] E. Liotou, T. Hoßfeld, C. Moldovan, F. Metzger, D. Tsolkas, and N. Passas, "Enriching http adaptive streaming with context awareness: A tunnel case study," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[139] 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.

[140] S. Tavakoli, "Subjective QoE analysis of HTTP adaptive streaming applications," Ph.D. dissertation, Telecomunicacion, 2015.

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

[142] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby, "User experience modeling for dash video," in *2013 20th International Packet Video Workshop*, Dec 2013, pp. 1–8.

[143] 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 Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 339–350.

[144] P. Juluri, V. Tamarapalli, and D. Medhi, “Measurement of quality of experience of video-on-demand services: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[145] K. Yamagishi and T. Hayashi, “Parametric packet-layer model for monitoring video quality of IPTV services,” in *2008 IEEE International Conference on Communications*. IEEE, 2008, pp. 110–114.

[146] Y. Qi and M. Dai, “The effect of frame freezing and frame skipping on video quality,” in *2006 International Conference on Intelligent Information Hiding and Multimedia*. IEEE, 2006, pp. 423–426.

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

[148] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, “Initial delay vs. interruptions: between the devil and the deep blue sea,” in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012, pp. 1–6.

[149] 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*, vol. 2014, no. 1, pp. 1–15, 2014.

[150] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, “Understanding the impact of video quality on user engagement,” in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM ’11. ACM, 2011, pp. 362–373.

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

[152] M. Zink, J. Schmitt, and R. Steinmetz, "Retransmission scheduling in layered video caches," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 4, 2002, pp. 2474–2478 vol.4.

[153] M. Dale, "Systems analysis and ecology," *Ecology*, vol. 51, no. 1, pp. 2–16, 1970.

[154] J. Pastor, *Mathematical ecology of populations and ecosystems.* John Wiley & Sons, 2011.

[155] J. Baranyi and T. A. Roberts, "A dynamic approach to predicting bacterial growth in food," *International journal of food microbiology*, vol. 23, no. 3-4, pp. 277–294, 1994.

[156] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement. correspondance mathématique et physique publiée par a," *Quetelet*, vol. 10, pp. 113–121, 1838.

[157] M. B. Schaefer, "Some aspects of the dynamics of populations important to the management of the commercial marine fisheries," *Inter-American Tropical Tuna Commission Bulletin*, vol. 1, no. 2, pp. 23–56, 1954.

[158] B. Gompertz, "On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies," *Philosophical transactions of the Royal Society of London*, vol. 115, pp. 513–583, 1825.

[159] S. Lederer, C. Muller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proc. of the MMsys*, 2012, pp. 89–94.

[160] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *NOSSDAV*, 2013, pp. 19–24.

[161] M.-N. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnstrom, and A. Raake, "Quality of experience and http adaptive streaming: A review of

subjective studies," in *Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on.* IEEE, 2014, pp. 141–146.

[162] 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)*, June 2013, pp. 682–686.

[163] E. Karapanos, J. Zimmerman, J. Forlizzi, and J.-B. Martens, "Measuring the dynamics of remembered experience over time," *Interacting with Computers*, vol. 22, no. 5, pp. 328–335, 2010.

[164] V. Seferidis, M. Ghanbari, and D. E. Pearson, "Forgiveness effect in subjective assessment of packet video," *Electronics Letters*, vol. 28, no. 21, pp. 2013–2014, Oct 1992.

[165] "geopy 1.11.0," https://pypi.python.org/pypi/geopy/1.11.0, Jan. 2016.

[166] C. Müller and C. Timmerer, "A vlc media player plugin enabling dynamic adaptive streaming over http," in *Proceedings of the 19th ACM international conference on Multimedia.* ACM, 2011, pp. 723–726.

[167] "Exoplayer," https://google.github.io/ExoPlayer/, Jan. 2016.

[168] "A reference client implementation for the playback of mpeg dash via javascript and compliant browsers." https://github.com/Dash-Industry-Forum/dash.js, Jan. 2017.

[169] R. Trestian, A.-N. Moldovan, O. Ormond, and G. Muntean, "Energy consumption analysis of video streaming to android mobile devices," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 444–452.

[170] M. Hosseini, J. Peters, and S. Shirmohammadi, "Energy-budget-compliant adaptive 3D texture streaming in mobile games," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys 13, 2013, pp. 1–11.

[171] M. A. Hoque, M. Siekkinen, J. K. Nurminen, and M. Aalto, "Dissecting mobile video services: An energy consumption perspective," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a.* IEEE, 2013, pp. 1–11.

[172] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is energy consumed in smartphone display applications?" in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '13, 2013, pp. 3:1–3:6.

[173] S. Khan, D. Schroeder, A. El Essaili, and E. Steinbach, "Energy-efficient and qoe-driven adaptive HTTP streaming over lte," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, April 2014, pp. 2354–2359.

[174] S. Kim, H. Oh, and C. Kim, "ePF-DASH: Energy-efficient prefetching based dynamic adaptive streaming over HTTP," in *Big Data and Smart Computing (BigComp), 2015 International Conference on*, Feb. 2015, pp. 124–129.

[175] MPEG, "Information technology – Dynamic adaptive streaming over HTTP (DASH) – part 5: Server and network assisted dash (SAND) ISO//IEC 23009-5," Tech. Rep., march 2017.

[176] Y. Shuai, G. Petrovic, and T. Herfet, "Server-driven rate control for adaptive video streaming using virtual client buffers," in *Consumer Electronics Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on.* IEEE, 2014, pp. 45–49.

[177] S. Wilk, D. Stohr, and W. Effelsberg, "VAS: A video adaptation service to support mobile video," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '15. ACM, 2015, pp. 37–42.

[178] S. Wilk and W. Effelsberg, "The content-aware video adaptation service for mobile devices," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. ACM, 2016, pp. 39:1–39:4.

[179] J. Hofbauer, K. Sigmund *et al.*, *The theory of evolution and dynamical systems: mathematical aspects of selection*, 1988.