Adaptive Bitrate Selection: A Survey

Yusuf Sani, Andreas Mauthe, and Christopher Edwards

***Abstract*—HTTP adaptive streaming (HAS) is the most recent attempt regarding video quality adaptation. It enables cheap and easy to implement streaming technology without the need for a dedicated infrastructure. By using a combination of TCP and HTTP it has the advantage of reusing all the existing technologies designed for ordinary web. Equally important is that HAS traffic passes through firewalls and works well when NAT is deployed. The rate adaptation controller of HAS, commonly called adaptive bitrate selection (ABR), is currently receiving a lot of attention from both industry and academia. However, most of the research efforts concentrate on a specific aspect or a particular methodology without considering the overall context. This paper presents a comprehensive survey of the most significant research activities in the area of client-side HTTP-based adaptive video streaming. It starts by decomposing the ABR module into three subcomponents, namely: resource estimation function, chunk request scheduling, and adaptation module. Each subcomponent encapsulates a particular function that is vital to the operation of an ABR scheme. A review of each of the subcomponents and how they interact with each other is presented. Furthermore, those external factors that are known to have a direct impact on the performance of an ABR module, such as content nature, CDN, and context, are discussed. In conclusion, this paper provides an extensive reference for further research in the field.**

***Index Terms*—HTTP adaptive streaming, adaptive bitrate selection, rate adaptation.**

I. Introduction

VIDEO streaming over data networks has been addressed as a research topic since the 1980s. In the early 1990s, video started to be transmitted over the Internet [1]. Since then, both the quality of the content and the variety of the video services have continued to grow. Nowadays, video is the most popular service on the Internet [2]. Cisco predicts that by 2019 global video consumption will account for 80%-90% of the entire data traffic traversing the Internet [3].

A typical video streaming service must accommodate a heterogeneous set of requirements due to the variety of contents and content sources, user contexts and interests, devices, and network limitations etc. Several video delivery schemes have been developed that can be categorised according to the type of network management used for the set-up and transmission of the video, i.e., a video delivery service is either implemented over managed or unmanaged networks [4]. Managed services are provided over a dedicated network infrastructure[1] that ensures Quality of Service (QoS). For instance, provided through techniques such as DiffServ [5], [6]. A typical example of the managed video service is Television over the Internet (IPTV), which is defined by the ITU-T as 'multimedia services such as television/video/ audio/text/graphics/data delivered over IP based networks managed to provide the required level of QoS/QoE, security, interactivity and reliability'. The video quality of the managed services is usually high[2] [10]. Nevertheless, managed networks are expensive to setup and maintain. Hence, those applications that depend on it, are typically provided by big organisations such as ISP, Telecom, or cable companies. Video services that are delivered over unmanaged networks are called *over-the-top* (OTT) services. The unmanaged services are mostly delivered to the end users via the best-effort Internet either by the content providers directly or through third parties. Since OTT services require no specialised or dedicated infrastructure their set-up and maintenance costs are relatively low. Another way of categorising video streaming service is whether it is *live* or *on-demand*. A live streaming service operates over events that 'take place over a defined time interval with defined start and end time'. As such, a user has no control over the session [11]. However, video on demand (VoD) provides a user with a stored pre-recorded content. This gives users the power to select which content they like, and watch it when the like [11].

Initially, it was assumed that video applications would not be able to achieve a good enough performance over the best-effort Internet because video transmission is a bandwidth intensive exercise and has a strict timing requirement. This has motivated research on quality assured video delivery services based on specialised network architectures and protocols [12], [13]. However, these efforts have not resulted in a wider deployment within the standard networks. Possibly, because of their complexity; the increase in capacity, as well as the wider penetration of broadband; improvement in the efficiency of the video compression techniques; and the prevalence of adaptive video access and delivery mechanisms. Today, video is usually streamed over the Internet without any change in the way the Internet classically works.

When streaming video over the best effort Internet, either TCP or UDP has to be used as the transport layer protocol. TCP is designed to ensure a byte-oriented reliable service. Additionally, TCP retransmits lost or corrupted packets and

Manuscript received May 29, 2016; revised January 26, 2017 and April 12, 2017; accepted June 19, 2017. Date of publication July 12, 2017; date of current version November 21, 2017. The work of Y. Sani was supported by Petroleum Technology Development Fund, Nigeria. *(Corresponding author: Yusuf Sani.)*

The authors are with the School of Computing and Communications, InfoLab2l, Lancaster University, Lancaster LA1 4WA, U.K. (e-mail: y.sani@lancaster.ac.uk; a.mauthe@lancaster.ac.uk; c.edwards@lancaster.ac.uk).

Digital Object Identifier 10.1109/COMST.2017.2725241

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

[2]For more details on the IPTV service see [7]–[9].

has an inbuilt congestion control mechanism, which allows it to ensure that a fast client does not overwhelm a slower server [14]. However, these protocol features result in an increase in delay and jitter. For video streaming, this results in frames arriving late, causing video freeze or rebuffering. On the other hand, UDP is a connectionless protocol that provides unreliable delivery service. It has no flow control mechanism, hence suitable for applications that put more emphasis on promptness over reliability. However, UDP is prone to packet loss. On the other hand, modern video compression techniques, such as H.264 [15], rely on compensation prediction algorithms, which in turn rely on the interdependence between successive frames. Imagine a lost in I-frame, it will be difficult for a player to successfully reconstruct the affected group-of-picture. Therefore, for UDP-based video streaming packet loss results in a degradation of the visual fidelity of the video.

The choice of transport layer protocol depends on the trade-off between visual degradation and video stall [16]. This fact notwithstanding, for OTT services research has shown that if the available TCP throughput is twice the bitrate of the media the negative impact of the protocol induced impairments can be ameliorated [17]. In fact, it has been shown that 'given any bottleneck bandwidth' OTT video streaming with TCP outperform UDP in terms of visual fidelity [16], [18]. Other advantages of TCP are: firewalls do not block its traffic, and it works well when NAT is deployed [19], [20]. Hence, TCP has developed into the dominant video transport protocol for the OTT services [21].

There are still challenges, especially considering the growing video resolution, and the resulting increase in bandwidth requirements. According to [22], a 720p video is encoded at an average of 2.5-3.5 Mbps, a 1080p at 5-6 Mbps and a 4K within a range of 18-20 Mbps—all having 30fps and using H.264 [15]. Obviously, provisioning bandwidth twice these encoding rates as suggested by the work of Wang *et al.* [17], especially in wireless environments where bandwidth is both scarce and expensive, is economically infeasible and often impractical considering current network infrastructures. One solution is to improve the compression efficiency as proposed in [23]. However, even when H.265 becomes widely available we can only expect about 40%-45% improvement in the compression efficiency [24].

The second approach, which is the focus of this paper, is to tailor the video quality to the device and network context. To achieve this, an adaptation mechanism is needed that will allow applications to gracefully adapt to the changing environment. To do this, the adaptation logic can be either located at the server-side [25]–[27], at the client-side [28], [29], or somewhere in-between [30], [31]. Server-side protocols are not considered scalable, though they might result in better network utilisation [11]. Client-side protocols are considered as being scalable and can be implemented using commodity servers. This makes them cheaper both financially and in terms of implementation complexity. However, client-side mechanisms are known to result in poorer network utilisation [11].

HTTP Adaptive Streaming (HAS) [32] is currently one of the most prominent video quality adaptation mechanisms, which is considered cheap and easy to implement [33]. Using HTTP over TCP provides additional incentives for adopting this technology. For example, HAS can be built on top of the existing content delivery technologies designed for ordinary Web usage [34]. In its standard form, HAS divides a video file into a number of chunks. Each chunk is encoded into multiple video rates and stored together with a description file called Media Presentation Description (MPD) [35]. A client continuously monitors and estimates its capabilities. It then requests a chunk with the highest video rate that is sustainable given the estimated capacity. The process, through which a client decides the profile and schedule of a chunk to download, is called *Adaptive Bitrate Selection* (ABR).

The first generation of ABR techniques strictly relied on throughput estimation for the video rate selection decisions [19], [36]. More recently, throughput based mechanisms have been enhanced and other parameters such as buffer occupancy, power level, cost etc. are also being used in the decision process [37], [38]. All these different approaches try to satisfy a set of general requirements, such as:

- Ensuring video is streamed with minimal number rebuffering events [39], [40].
- Maximising both the minimum and the average video quality level [41]–[43].
- Ensuring a consistent experience by minimising the number of quality level switches [44]–[46].

To fulfil these requirements, or any other objective set by the ABR designer, such as fairness [47], power saving (see Section III-C for more detail) and others, the various subcomponents of an ABR system must not only work well individually but also have to collectively function as a well-defined unit. At the same, the ABR scheme must be designed in such way that it takes advantage of the specificity of the context it operates in. Currently, most research only looks at a particular aspect or methodology and do not sufficiently consider the overall context [48]. For instance, research has investigated aspects such as improved bandwidth estimation [49], [50], buffer management [38], energy consumption [51], adaptation logic [37], [50] among others. This paper takes a more comprehensive approach and argues that to understand the video quality adaptation problem in HAS, research needs to pay attention to all the relevant aspects of ABR. Some of surveys related to HAS exist. For example, in [52] a review of the past two decades of video streaming, which includes HAS, are presented. In [35] and [53] a brief survey of some existing HAS standards and implementations are presented. Some other surveys that focus on QoE issues related to HAS are [54]–[57]. At least to the best of our knowledge, at the point of writing this paper, no survey exclusively focusing on the ABR scheme currently exist.

This paper presents a comprehensive survey of the *client-side video quality adaptation mechanism for a video on demand service delivered over the best effort Internet, and the* *related issues*.[3] It starts by dividing the factors affecting ABR system into two classes, namely, external and internal factors. A taxonomy of the factors that influence the performance of ABR is presented in Fig. 1. The external factors are represented by aspects that are exogenous to the ABR algorithm and are known to have an impact on the performance of the ABR scheme. And the internal factors are composed of those elements that an ABR designer has complete control over.

First, how ABR internally operates is presented by initially introducing a framework that decomposes ABR functionally into three subcomponents, and then discusses the relevant interfaces between them. The three components are *resource estimation, chunk scheduling and adaptation module*. Subsequently, factors related to the content nature (e.g., chunk size and the encoding used - AVC or SVC), CDN, and operating context are presented. The objective is to understand how external factors affect the working of a typical ABR algorithm. The contributions of this paper are as follows:

1) The paper presents a review of the adaptive bitrate selection problem, as a whole, in an accessible form for those not familiar with the field.
2) It also presents a review of the individual components of ABR scheme addressed by the research community in a self-contained manner, which is designed to help researchers get a better understanding of a specific aspect in relation to the overall context.

The rest of the paper is arranged as follows. Section II presents an overview of the HAS system and ABR framework used throughout the paper. Section III covers resource estimation and Section IV introduces a review of different techniques for scheduling chunk requests. In Section V the adaptation function is detailed. While Section VI presents a discussion on how these various components interact each other, with Section VII discussing the impact of chunk preparation on the effectiveness of ABR. In Section VIII the impact of operating context on the effectiveness of an ABR scheme is talked about. In Section IX CDN and its impact on ABR is covered. Section X presents a summary of the most important lessons learnt, and some of the challenges facing the community. Finally, a summary and conclusions are presented in Section XI.

II. ADAPTIVE STREAMING

A. Overview of HTTP Adaptive Steaming

HTTP Adaptive Streaming (HAS) [47] allows content providers to cater for the requirements of multitude devices and different contexts. It was first introduced by Move Networks in 2007 [58], [59]. Presently, almost all the standardisation bodies that have an interest in media delivery over the Internet have either separately or jointly standardised it: IEFT [47], 3GPP [60], MPEG [32] and the Open IPTV Forum (OIPF) [61]. Further, major IT companies also provide implementations or versions of it as part of their products, e.g., Microsoft Smooth Streaming (MSS) [33], Apple's HTTP live Streaming (HLS) [47] and Adobe OSMF [62]. Additionally, many commercial content providers are increasingly adopting it, for example, Netflix [38] and YouTube [63].

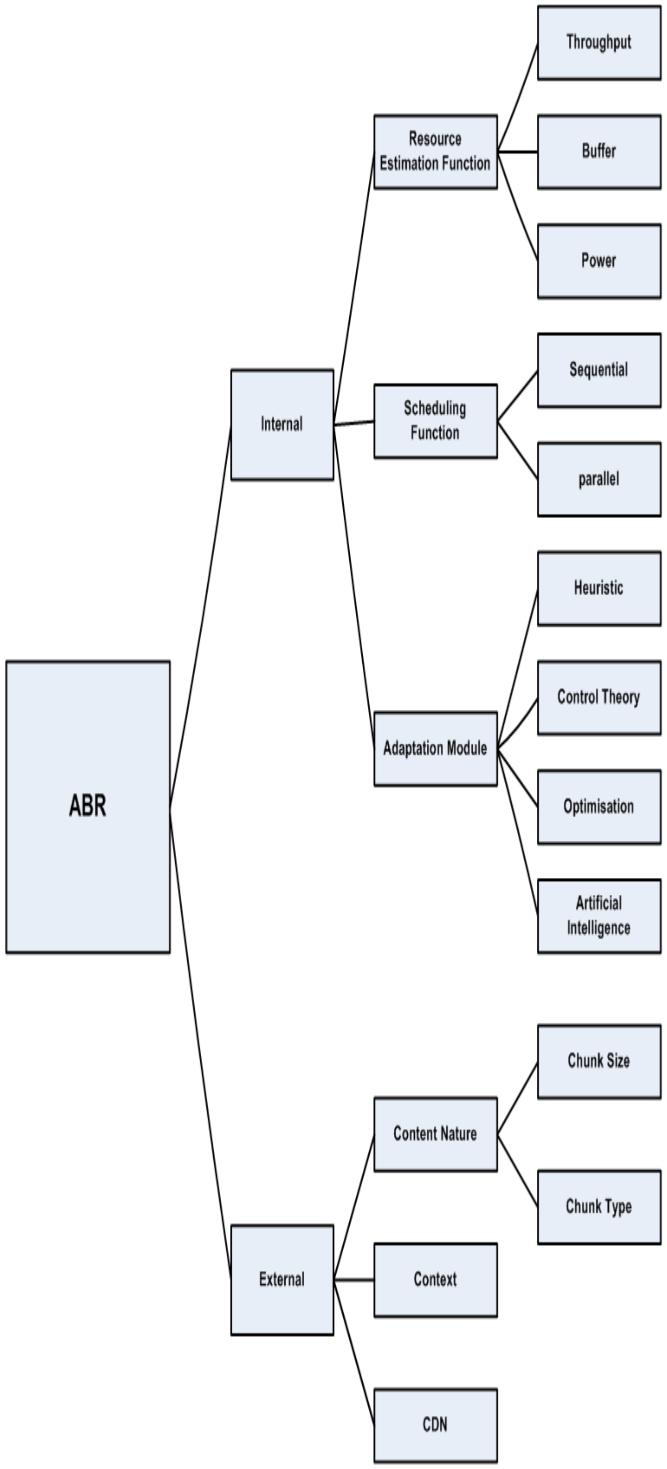


Fig. 1. Taxonomy of the Factors Influencing the Performance of ABR as Used in the Paper.

[3]It should be noted that out of necessity we consider some aspects of either server-side or network-assisted ABR. However, this is only done when it is very clear that they directly enhance the efficiency of the client-side function and not act as self-contained independent protocols. This restriction is by no means intended to reduce the importance of these paradigms, but rather because we believe that they are now matured and broad enough to deserve an independent attention.

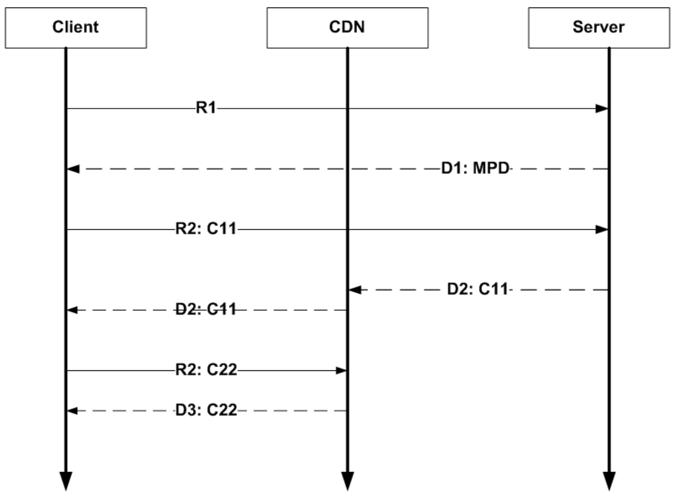


Fig. 2. HAS Event Sequence Diagram.

In its standard form, HAS divides a media file into fragments, conventionally of equal size in terms of playtime.[4] Each fragment is called a *chunk* or *segment* (this paper uses both interchangeably). Each chunk is encoded in multiple video rates to satisfy the requirements of different devices and network conditions. The criteria for providing an optimal set of video quality representations are discussed in detail in [64]. The chunks are all stored together with a description file called *Media Presentation Description* (MPD) [65] on one or more servers (which can be any Web server). MPD is an XML metadata file containing a description of the available chunks. It usually provides information such as the number of different video rates per chunk and the duration of each chunk in seconds. As can be seen in Fig. 2, when a client first requests a video file, e.g., using an HTTP GET request, the server responds by sending an MPD. The client will then use the information contained in it to construct the Uniform Resource Identifier (URI) for the subsequent requests.[5] The parsing and processing of the MPD are handled by the media presentation module. The detail of MPD is outside the scope of this paper but has been well documented elsewhere (for more detail see [47], [60], [65], [66]). After the MPD has been received and the subsequent construction of the URI, a client progressively sends requests for the next available chunk until the end of the video. The schedule and the video rate of each of the requested segment are purely decided based on the client's estimation of its available system and network resources. For example, Fig. 2 shows how for each subsequent request a client specifies both the sequence number and the quality level. In this example, R2 requests C22, which stands for a chunk number two with the second highest quality level.

The number of response-request between client and server in HAS depends on the chunk size, in fact, it was found, in [67], there is an inverse proportional relationship between them. In a delay intolerant service, such as live streaming service, this can result in a degraded experience. To solve this problem schemes that employ HTTP 2.0 server push capabilities have been proposed [67]–[69]. A server either pushes k number of chunks after the receipt of a request from a HAS client or continuously sends chunks back-to-back until requested otherwise by a client [68]. However, in their subsequent work [70] they found the k-push scheme negatively affecting network adaptability and wastes network resource by over-pushing chunks which may be abandon by user. They then propose a scheme that dynamically adapts the value of k depending on the client context. Their result shows the adaptive-push scheme has alleviated the earlier reported issues.

One important aspect of HAS as can be seen from Fig. 2 is its ability to seamlessly work with Content Distribution Networks (CDN) or any proxy server technology used by standard HTTP services. This means a content provider can cache its video content on a third party CDN infrastructure to save costs and reduce download latency [71], [72]. This should normally be transparent to clients. A client only makes an HTTP GET request, whereas it is then 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.

B. Adaptive Bitrate Selection

A typical HAS system is divided into two different parts, that is, media presentation and the Adaptive Bitrate Selection ABR. While the media presentation module of HAS has been successfully standardised [33], [60], how the actual adaptation logic works is left to the individual developers to decide. This open nature is intended to encourage innovation.

In order to adaptive video quality to a context, network and other system performance parameters such as CPU, display size or battery life are measured. The choice of which parameter becomes a situational indicator depends on the QoE metric that an ABR intends to optimise. The ABR then uses the measurement result in making a decision on the schedule and profile of the chunks to be downloaded.

A typical ABR scheme operates in two states [36], [73], [74]: ***Buffering*** and ***Steady*** state. At the buffering state, conventionally, a player starts requesting the lowest video rate. Thereafter, the ABR may try to fill its buffer as quickly as possible [38], [75], raise the video quality as fast as possible [36], [73], or combine the two approaches [29]. 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 [74]. However, throughput based ABRs (see Section III-A for a detailed discussion on throughput based ABRs) assume that a HAS system can sustain a video rate up to the measured throughput. Therefore, try to match the video rate, as close as possible, to the estimated network capacity, and hence from the start of streaming such approaches continuously raise the video rate. In [29] and [76], a video rate is increased non-linearly, such that after every chunk request the video rate is increased by a factor until the sustainable bitrate is reached. The approach gradually raises the quality level

[4]In Section VII-B a detailed discussion is presented on variable chunk sizes and their impact on the performance of an ABR scheme.

[5]Note, the URI may be directly provided or has to be constructed from a template.

without increasing the risk of buffer depletion. 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. 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 enough content in the buffer. This can be achieved when either a certain amount of content is downloaded or the buffer size reaches a predefined target [77]. For example, the MSS has a playback buffer of about 20 seconds but starts playing when the buffer contains just about 10 second worth of download [36], [77]. Likewise, Netflix has a buffer size of 300s worth of content but begins playback 13s after receiving the first packet [36]. The duration of the buffering phase is called ***convergence time***.

When the buffer size reaches a particular threshold, the steady state mode is activated. In this phase, the goal is to maximise the video rate of the requested chunks and ensure buffer over-flow does not occur. It should be noted that the buffer contains a sufficient amount of content to absorb network instability and the chances of under-run are low. Therefore, only chunks having the highest video rate the available system capacity can sustain are requested. One important feature of the steady state phase is called ***periodic download***. The 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.[6] The periodic download guarantees an inflow of content into the playback buffer without causing a buffer overflow.

To better manage the complexity of ABR, Jiang *et al.* [49] propose a general framework. The framework divides an ABR module into three subcomponents, each unit is associated with a function that an ABR is expected to render. It is worth noting that the subcomponents need not to be necessarily co-located. In fact, any of the modules can be located on separate systems. However, in this paper, we will concentrate on the case where all the subcomponents are co-located on the client device. Here are the subcomponents:

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

As can be seen in Fig. 3, the chunk scheduling function takes as inputs the time the last chunk download finished in conjunction with the buffer level and is responsible for deciding when a chuck is going to be requested. The adaptation module decides which profile of the chunk should be downloaded, based on feedback from the resource estimation module. An obvious shortcoming of this framework is that it treats the ABR module as if it exist in isolation of its context. Simply put, it does not consider the external factors affecting the performance of an ABR. In this paper, we present an improved framework that takes a system view of the ABR module. As shown in Fig. 4, this framework explicitly

[6]Extensive experiments have confirmed the presence of this ON-OFF traffic characteristic in both commercial and experimental players [36], [49].

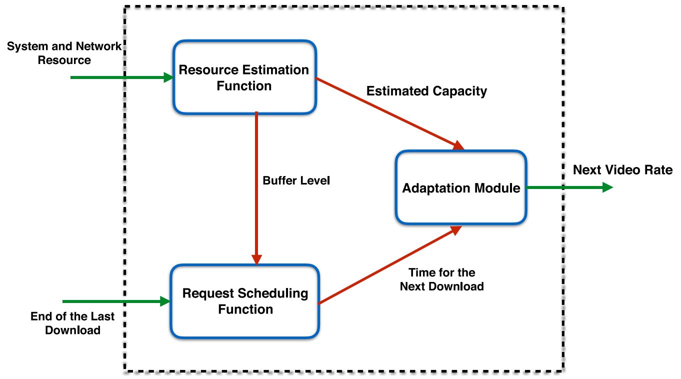

System and Network Resource
Resource Estimation Function
Estimated Capacity
Buffer Level
Adaptation Module
Next Video Rate
Time for the Next Download
Request Scheduling Function
End of the Last Download

Fig. 3. ABR Framework.

considers external factors such as delivery and content format, context or user requirements.

A lot of effort has been put into understanding both the impact of the interaction between the internal components and the influence of the external component (i.e., operating environment) on the performance of HAS services. The research in this spaces addresses both, theoretical and experimental aspects, which includes several detailed performance evaluations of some of the existing services. For instance, in [36], a performance analysis is conducted to determine the impact of throughput fluctuation on the performance of some of the most prominent commercial implementations of HAS services. Furthermore, Akhshabi *et al.* [42] investigate the impact of competition for the available bandwidth between multiple adaptive streaming clients. In a similar vein, the work discussed in [78] presents a comprehensive evaluation of the impact of competing flows on a typical adaptive streaming player. Other researchers [74], [79] have looked into the effect of the scheduling policy. However, Timmerer *et al.* [80] investigated what benefit, if any, is obtained by the use of different adaptation mechanisms. How to properly select the representation set that is presented to clients is the topic of the research outlined in [81]. While this is not an exhaustive list, it summarises the major research related to theoretical and experimental performance evaluation on how the different internal components impacted on one another and their external environment. However, more performance evaluation work will be introduced in the relevant sections throughout the paper.

III. RESOURCES ESTIMATION FUNCTION

Resources and their availability define the capabilities of a HAS client. Therefore, it is crucial to understand the way they are monitored and measured, and how they affect the efficiency of an adaptation module.

Typically, a resource estimation function is expected to monitor and measure the resource of interest. This task can be implemented at the server-side, the client-side, or somewhere inbetween. The appropriate location depends on which performance parameter an ABR scheme relies on. The client-side ABRs distribute the task of resource observation, hence are more scalable than server-side implementations.

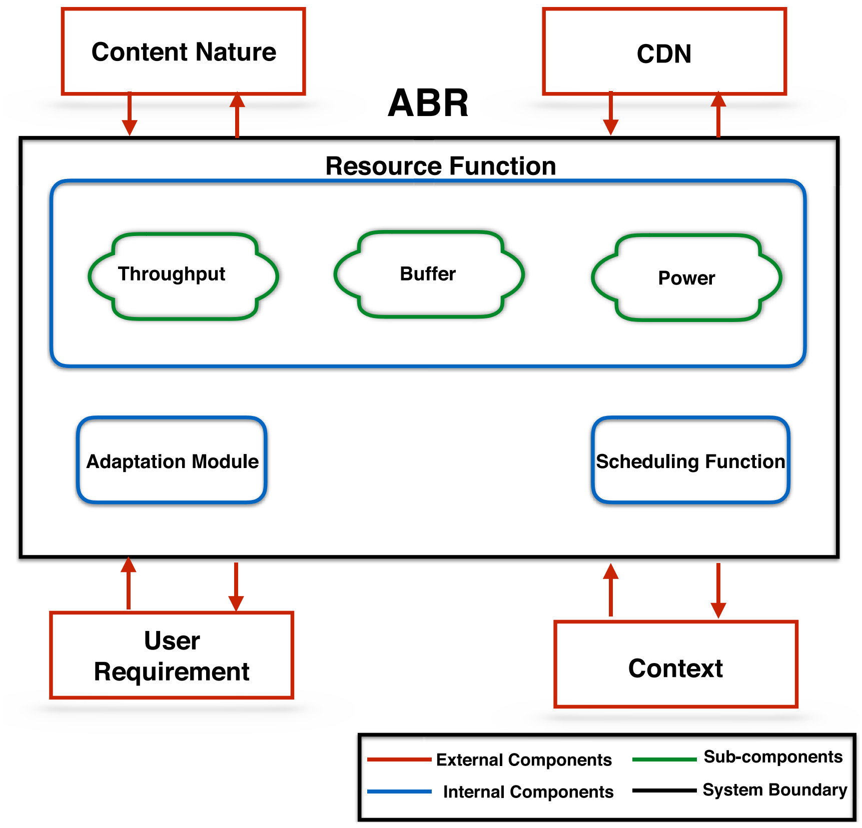


Fig. 4. System Representation of ABR Module.

Because content providers target a large number of clients, and on top of this, most of the important parameters may be better observed closer to the client (e.g., last-mile bandwidth, buffer occupancy and power level), monitoring and resource estimation is usually implemented at the client-side. However, solely relying on client-side observations results in an *opportunistic* ABR. In other words, an ABR that only optimises the performance of a single client [40]. To address this challenge, a central control plane has been proposed in [82], which aggregates measurements from many clients. This ensures that ABR scheme is globally optimising the performance across all clients. In [83], a control plane was used to orchestrate the monitoring and measurement process of video streams in a network. The goal here is to ensure network-wide QoE fairness. Cofano *et al.* [30] also propose a network control plane with the goal of maximising network-wide QoE and bandwidth utilisation. They rely on bandwidth reservation on a per-flow basis to achieve this. Also [31] propose a software defined networking (SDN) based architecture that allocates bandwidth to competing clients based on both content complexity of requested video and the buffer status of individual clients.

Selecting which resource is used as a situational indicator in video rate adaptation is context dependent [28]. The first generation of ABRs principally relied on throughput estimation and always selected the highest video rate lower than the measured throughput [36], [87]. It was assumed that this strategy can avoid rebuffering while at the same time providing high quality video. It later became obvious that throughput estimation alone is not a sufficient parameter for designing efficient ABR scheme [49], [78]. This is due to the inadequacy of the network state information to capture the requirements of the variety of devices and the different contexts that HAS is expected to operate under. For example, a user with limited battery power watching European Champions League finals using his smartphone perhaps will be more concerned with his battery life (in order to prolong his viewing) than the quality of the video. In this context, it may be more appropriate for an ABR scheme to make a decision based on power consumption rather than aiming at delivering the highest video rate. Nowadays, various parameters are used to adapt video to the multitude of requirements of different clients. Table I presents a summary of the parameters and the respective work that have

TABLE I
CLASSIFICATION OF ABR BASED ON THE RESOURCE DEPENDENCE

Resource	Research Work
Throughput	[19], [28], [37], [49], [50], [59], [77], [84], [85], [86], [87], [88], [89], [90], [91], [79], [92], [93], [94], [95], [96], [97], [98]
Buffer	[28], [37], [38], [99], [100], [101], [78], [75]
Power	[102], [103], [104], [105], [106], [51], [107], [108], [109], [108] , [110], [109], [111], [112], [113]

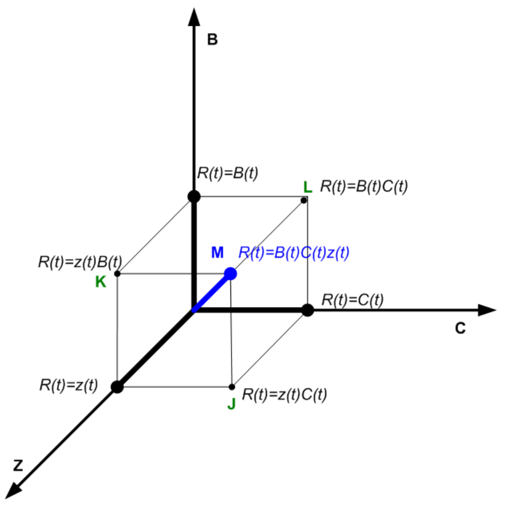


Fig. 5. The set of all possible representations of $R(t)$. The $C(t)$,$B(t)$ and $Z(t)$ axes represent $R(t)$s that rely on one metric. While points J, K, and L represent $R(t)$s with one main factor and one adjustment factor. M is $R(t)$ that works with three and above factors.

used them as factors in deciding the most appropriate video rate to stream.

For the purpose of resource utilisation, ABR can be represented as a function $R(t)$ according to which the video rate is selected. Fig. 5 captures the different representations of $R(t)$ that are theoretically possible. A typical $R(t)$ takes various parameters as inputs, for example, throughput [49], buffer occupancy [38], power level [51] and cost [113], each weighted depending on the desired outcome. The principal parameter an ABR relies on is called the ***main factor*** with weight γ, other factors are then ***adjustment factors***, with a combined weight of $(1 - \gamma)$. When no parameter dominates, all should be considered as adjustment factors. $C(t)$ 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).

The $C(t)$, $B(t)$ and $Z(t)$ axes represent $R(t)$s relying on one parameter only. 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) while the 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 parameters 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 parameters, for instance as discussed in [86]. 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 uses 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. Researchers have mainly focused on cases where $C(t)$, $B(t)$ and $P(t)$ are used either as the main or adjustment factor. Thus, in the following, the paper will also concentrate on these cases.

A. Throughput Based ABRs

Throughput-based ABRs try to estimate the available network capacity, which is the average unutilised capacity over a specific time interval [114]. However, regardless of the underlying technology or the transport protocol used for any content transmission, the available bandwidth is time-varying [115]. 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).

Usually, throughput-based ABRs equate the available bandwidth with the measured TCP throughput. Furthermore, the monitoring and the estimation of the available network capacity are often done above the HTTP layer, which results in not very accurate information [78]. 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. However, the instant throughput derived from a single chunk is hardly used since it is prone to short-term fluctuations as result of, for instance, the time-varying nature of the available bandwidth, or the dynamics of TCP. Regardless of the cause of fluctuations, the per-chunk estimated throughput can cause significant variability in the video quality. Due to this and the difficulty in accurately estimating throughput above the HTTP layer various techniques are used to improve the quality of the measurement. Table II gives a summary of the techniques and the corresponding proposed solutions.

1) Estimation Techniques: In order to overcome the problems related to per-chunk estimation and the instability this creates, a variety of methods has been used to estimate the available bandwidth. In [19], μ is multiplied by the video rate of the currently received chunk, where μ is the quotient of Media Segment Duration (MSD) and Segment Fetch Time (SFT). The SFT is the time the last bit of a requested chunk is received minus the instant of sending the request. Where MSD is the media playtime contained in a chunk. And the result is used as the estimated throughput. However, the algorithm relying on this estimate was found to lead to oscillatory video quality. In their follow-up work [84] they proposed a method for finding an optimum value of MSD involving a number of chunks. Simulation results showed that using the optimum value of MSD can smooth-out the oscillation previously observed in [19].

Long before HAS was proposed, Prasad *et al.* [115] argued that any meaningful available bandwidth estimation technique requires a time averaging of the instantaneous estimates over a time interval. Several ABR research has used different types of averaging techniques in estimating the available throughput. Akhshabi *et al.* [36] experimentally evaluated some players. For MSS, they found that it is using a kind of a running

TABLE II
SUMMARY OF THE VARIOUS EFFORT TO IMPROVE THE THROUGHPUT ESTIMATION

Dimensions of research	Main findings	Solutions
Estimate Techniques	-Instant throughput results in video quality fluctuations.	-Averaging of instantaneous estimates is needed.
Reliability of Estimate	-Accurate bandwidth estimation above the HTTP layer is hard. -Inaccurate estimation causes underutilisation and rate oscillation	-Improving the information from TCP to HTTP. -Using lager chunks size.
The Impact of TCP Dynamics	-TCP congestion control dynamics has an impact on video quality.	-TCP congestion control mechanism needs improvement. -Increasing video Chink size .
The Impact of Traffic Pattern	-The ON-OFF traffic pattern has an impact on video quality.	Delaying the OFF until after ramping-up phase.
Resource Pooling	-Resource pooling techniques -Increase the available bandwidth.	-The Use of multiple interfaces simultaneously.

average (MSS is a propriety system). This conclusion was reached since the player does not instantly react to a change in the available bandwidth. This is the case regardless of whether the player is stepping up or down the video quality level of the recently downloaded chunk.

In [49], 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.* [50] 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. Gouache *et al.* [85] compute the average and variance of the estimated throughput of a number of chunks using a low-pass filter (moving average). The current throughput is then obtained by subtracting the product of the variance and a constant from the calculated running average. The constant controls the conservativeness of the adaptation logic. They found the algorithm estimate close to the maximum capacity and is robust to network errors. To improve the stability of ABR that relies on this type of history-based throughput prediction, a safety factor is sometimes applied to the estimate, for example, as discussed by [77] and [86].

It is a well-known fact that smoothing techniques have a tendency of inhibiting the responsiveness of an algorithm. This may cause a late reaction to a significant variability that perhaps requires an urgent action. In the context of video rate adaptation, this may result in a late response to a large throughput decrease and subsequent video freeze. To remedy this lack of responsiveness, in [87] and [88] the use of an adaptive coefficient (weight) for the weighted moving average is proposed. The approach was found to increase the responsiveness of ABR without causing unnecessary video rebuffering.

2) Reliability of the Estimate: Still an open issue in ABR research is the extent to which the throughput estimates reflect the true state of the available bandwidth.

Any throughput measurement done at the application layer is at best only the throughput of the underlying TCP. However, Jain and Dovrolis argue that equating available bandwidth with 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 etc.) [114].

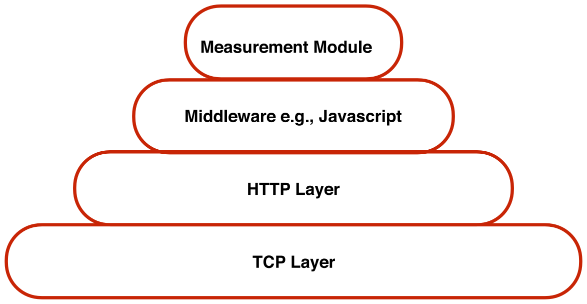


Fig. 6. HAS Protocol Stack.

Similarly, an argument against equating the TCP throughput observed at the application layer with the available bandwidth is presented in [59]. The paper has shown that when clients compete at the bottleneck link, the presence of a competing application and the discrete nature of the HAS downloads make it difficult for a player to estimate its fair share of the available bandwidth correctly. As a consequence, there is an under-subscription of the available bandwidth and video rate oscillation. The latter is known to negatively impact user experience [42]. As a remedy, the paper proposes 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. The paper argues that this happens when the network is congested, and the off-interval is absent. Otherwise, it probes the network by incrementing the sending rate and backing off when congestion is detected.

Fig. 6 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 in [78]. The paper investigated three video streaming services, that is, Hulu, Netflix and Vudu, and found that an accurate client-side bandwidth estimation above the HTTP layer is hard. The authors proceed to argue that any rate selection decision done on the of such an inaccurate estimate will trigger a feedback loop that leads to an undesirable variability and unnecessary reduction in video quality. Furthermore, they observe that the root cause is lack of information, because 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 suggests that an ABR scheme either improves the information flow from TCP to the HTTP layer to ensure that TCP has a chance to reach its steady-state (e.g., by increasing the segment size), or to just strictly rely on buffer state changes for video rate selection decisions.

Application layer based schemes are not the only techniques used in predicting the achievable throughput. Many attempts were made on cross-layer throughput estimation. In [37] machine learning technique developed in [116] is used to predict the achievable throughput. The method uses the support vector regress algorithm [117] to train the throughput prediction model with network layer information like packet loss, delay, and RTT. In [89] and [90], physical layer 'goodput' is used to complement the application layer estimate. This results in an improvement in the perceived video quality and a reduction in the rebuffering frequency.

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 also plays a significant role. The effect of TCP becomes more evident in the wireless context. This is because of the well-known fact that wireless communication systems are characterised by high latencies, high bit-error rates and protocols that have to cope with handling user mobility. The solution to this is either to modify the congestion control mechanism of TCP (as recommended in [79]) or to reduce the causes of the unnecessary activation of the congestion control mechanism. ABR researchers tend to prefer the latter option.

In [118], multiTCP is proposed, an application layer algorithm that improves resilience against short-term TCP throughput fluctuation. The authors of the paper demonstrate that for any single packet loss event, the reduction in throughput when two TCP connections are used is four times less than if one TCP connection only is used. In summary, the amount of TCP throughput reduction is inversely proportional to the number of TCP connections employed

In [92] multiple HTTP/TCP connections are used to stream HAS content. The scheme is found to be insensitive to packet loss and therefore, reduces throughput fluctuation. In brief, the scheme works as follows: it downloads multiple (but fixed number) of chunks from a server. Each stream requests chunks in the order of playback. But the chunks that are needed soon are prioritised. When a chunk stalls and reaches time-out, it is retransmitted by two HTTP streams. While in [92] and [118] all streams share a bottleneck link, in [85] an attempt to improve the resilience of the system by concurrently fetching chunks from multiple servers are 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.

4) Impact of Traffic Pattern: Apart from the capacity estimation performed at a wrong layer and the TCP dynamics. Another cause of the inability of a client to estimate its fair share of available bandwidth is nature of HAS traffic pattern.

Usually, an ABR requests chunks discretely. Furthermore, when the buffer reaches a threshold download ceases until a certain amount of content has been consumed to free up buffer space. This results in an *ON-OFF* traffic pattern, which is obviously bursty. Fine-tuning bursty traffic is a well-established technique used in many bandwidth estimation applications [119]. This is the case since most bandwidth estimation techniques work best under the assumption that the network traffic has fluid flow characteristics [95]. It should be noted that this assumption may not be always correct. Jain and Dovrolis [114] 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 a fluid flow. One technique used to achieve this is traffic shaping. Traffic can be shaped at the server, the gateway, or the client.

Akhshabi *et al.* [93] 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 extenuated 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 hence would have a reduced execution overhead on the server, the Web server still requires modification.

In [94], a traffic shaping mechanism implemented at the home gateway is proposed. 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. The authors are of the opinion that implementing a bandwidth manager at the gateway has a number of advantages. For example, the gateway has a global view of the network and can allocate bandwidth based on factors such as device roles, characteristics, and content value.

In [95], it is argued that traffic shaping at a gateway is not a good alternative since it involves the participation of a network operator, and may require additional functionalities and permissions that broadband operators are not always willing to provide. They propose a client-side traffic shaping technique that does not involve the server or the gateway in any way. The technique is based on their earlier work [96], where they used different segment durations to randomised chunk inter-arrival time (see Section IV). They interleave requests from different clients. When traffic consists of low profile chunks, the effect of shaping was 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.

5) Resource Pooling: Over-provisioning could be an option in dealing with issues related to bandwidth estimation above the TCP. However, this is not always possible since bandwidth is an expensive resource, and in some instances (e.g., in wireless environments) it can be quite scarce. Nonetheless, a number of researchers have attempted to improve the available bandwidth through various resource pooling techniques [120], whereby a collection of resources belonging to a system is treated as a single pool. The idea is to exploit the different communication resources a device has access to, e.g., today wireless devices are equipped with multiple network interfaces which can be easily aggregated.

Wang *et al.* [121] investigate a scenario where a client streams over multiple paths, which may or may not share a bottleneck link. Their goal is to determine under what conditions multipath TCP provide satisfactory performance, and what are the potential benefits of using multiple TCP connections. The results of an extensive simulation show that the use of multiple TCP connections provide satisfactory performance when the achievable aggregated throughput is 1.6 times the bitrate of the video rate with few seconds of start-up delay. This represents a reduction in the bandwidth requirements of a single TCP connection, which according to the authors' earlier work requires 2 times the video bitrate [17]. Additionally, they found that with proper design the aggregate throughput of the multiple paths can be equal to (or perhaps greater than) the sum of the throughput of the individual paths.

Also in [97] an attempt has been made to efficiently aggregate available bandwidth from multiple heterogeneous network interfaces. The authors use HTTP range retrieval requests to logically segment a chunk, then simultaneously download each logical chunk over the various interfaces. They found an increase in performance compared to a single link, and consequently a significant increase in video quality. However, to achieve an optimal performance there is a need for a large buffer size to compensate for link variability. In their subsequent work [98], the sub-segment approach is improved, so that chunks are dynamically segmented in proportion to the estimated link capacity. This avoids idle periods and large buffer requirements by allocating the right amount of data to each link.

B. Buffer-Based ABRs

Buffer serves many purposes in streaming systems. Initially, it is introduced to absorb throughput variability [122] so as to ensure that the time restrictions of continuous 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 under-run a certain amount of data is *buffered*. This ensures that the HAS client will continue playing from the pre-buffered content for at least a time equivalent to the duration of the buffered video. Therefore, there is an inverse relationship between buffer size and the probability of video under-runs, i.e., the bigger the size of a buffer the longer it takes to run out of content. However, in a practical system, there is a limit to the size of the buffer that can be used [123].

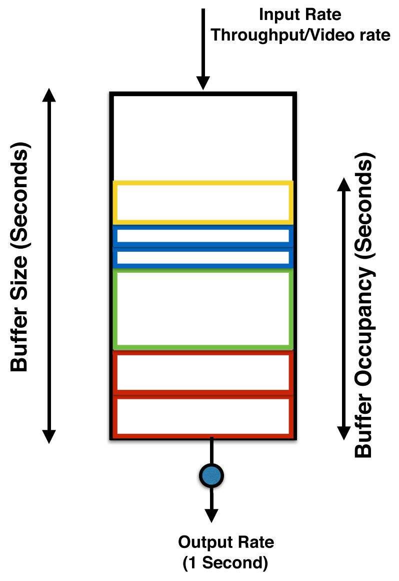


Fig. 7. Buffer Dynamics.

1) Buffer Dynamics: In the early days of video streaming, before scalable video or HAS video streaming services were available, a video 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.[7] 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 Fig. 7) at any given time can store multiple chunks of which each chunk can be of any video rate between the lowest and the highest available. Nonetheless, the video chunks usually are of equal size in terms of playtime. The relationship between the buffer size in terms of storage capacity (or spatial size) depends on several parameters such as the number of chunks buffered, their bitrates and the encoding scheme used (VBR or CBR) [75]. This complicates the process of converting the buffer occupancy in bytes to time. To the best of our knowledge, currently, there is no straightforward method of doing this conversion. Hence, buffer in HAS is generally calibrated in time [29], [37], [49]. This greatly simplifies the tracking of the buffer occupancy as a player only needs to multiply the temporal chunk size by the number of chunks contained in the buffer. This is further aided by the fact that memory is not a critical issue any more.

[7]Considering the fact that the variable bit rate coding is the most widely used encoding scheme. The average video playback rate is used.

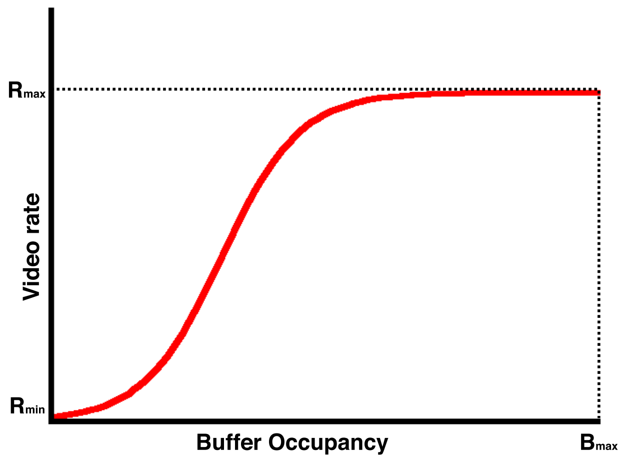


Fig. 8. Buffer Evolution Map.

In summary, 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 Fig. 7). 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 is greater than *one* (i.e., the play-out-rate) buffer occupancy grows, if it is less than *one* it shrinks. This effectively means that to maintain the buffer occupancy at a certain level, before a chunk is played the next chunk must have arrived. If the ABR continuously requests chunks that are above what the network can sustain the buffer depletes, and consequently there will be a playback freeze at one stage. Moreover, the lower the current buffer occupancy becomes the more likely it is that the video will freeze. In fact, Xu *et al.* [99] have found that the probability of buffer starvation decreases exponentially with respect to the initial buffer level. Thus, to improve a player's resilience against rebuffering events a client does not instantly start playing but waits for some time to fill the buffer until a threshold (also called *low watermark*) is reached. This period is called *start-up period*. It was observed in [124] that a large initial buffer occupancy increases a streaming application's tolerance to network variability but at the expenses of a start-up delay. However, users are not willing to wait longer than few seconds before the start of a playback [125]. In fact, it has been found in [126] that viewers start to abandon a streaming session if the start-up delay takes more than few seconds. To solve this trade-off, at the start of a streaming session ABR algorithms start requesting the chunks with low representation because downloading a low quality level results in fewer data, and hence reduced buffer requirement and faster content transfer.

2) Buffer As a Feedback Signal: Until recently, ABR algorithms usually divided buffer into logical segments $S_0, S_1, \ldots, S_{n-1}$ with $B_1 < B_2 < \cdots < B_{max}$ as thresholds, where a buffer occupation with low segment numbers (i.e., a segment number close to zero) indicates an increasing likelihood of buffer depletion. The least number a buffer can be segmented into is two. The first segment S_0 is an area from when the buffer contains no data to a threshold point B_1.[8]

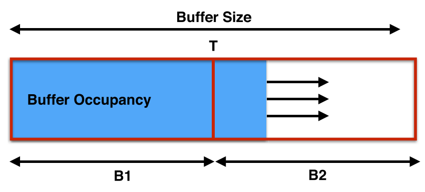
[8]Note, this may be any point less than the maximum buffer size.

Second segment S_1 is an area from the threshold to the maximum buffer level (B_{max}). An example is [37]. However, others such as [28] and [73] divide the playback buffer into three segments, called *panic*, *low*, and *upper* level. The logic behind the segmentation is to allow the pertinent ABR algorithms to behave differently at each buffer segment [28], [37], [127].

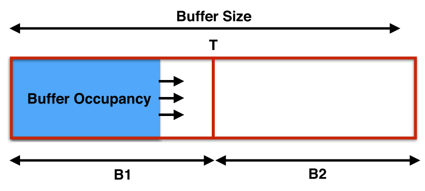
In [38], a video rate map that continuously maps the video rate to buffer occupancy is presented. It starts by separating the buffering phase (called *reservoir*), in which only the lowest available video rate is downloaded from the ramping-up period where the video quality is linearly incremented. However, Sani *et al.* [29], [76] argue that this creates a disconnected flow. Furthermore, the linear evolution of video quality level prolongs the convergence period. They propose a logistic rate evolution map that tries to unify all stages of streaming. In Fig. 8 a video quality evolution map [29], [76] is presented where the horizontal axis represents the buffer occupancy in seconds and the vertical axis shows the video rate (R_{min} and R_{max} being the minimum and maximum video rate respectively). As can be seen from the curve, the rate at which video quality changes relative to the buffer occupancy is non-linear. The amount of buffer needed to change the video rate at the start is relatively higher, then becomes increasingly smaller as the value of the video rate increases. Before finally it starts increasing again as the video rate approaches the maximum level. Their argument is that at the start of a streaming session ABR should be conservative with respect to quality change because the risk of video freeze is high. But when the buffer contains enough content to allay this fear the ABR can exponentially increase the video rate. Furthermore, the authors reasoned, from a point where the video quality level is relatively high a rise in the current rate does not necessarily translate into an equivalent improvement in the user-perceived quality [128]. Hence, the rate at which the video quality level changes is slowed down by increasing buffer needed to change to a high video rate. This not only helps raise the video rate to the maximum without any increase in the number of rebuffer events but also protects the system from video quality fluctuation.

In [129], the relationship between buffer state change and video rate is modelled using a bio-inspired approach. The paper assumed that the buffer is a habitat with a limited caring capacity, with the video rate being an animal species whose increase in population an ABR is interested in. Furthermore, they assume that the rate at which the content arrives is the birth rate and the rate at which a player consumes content is the death rate of the species. With this information, they used Verhulst-Pearl equation to represent the video rate map. Results of experiments show and improvement in the average value, and the stability of the video rate compared to [38].

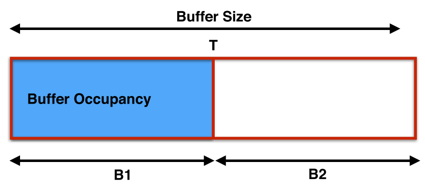
Another important feedback signal that can be derived from the buffer occupancy is the trend of the buffer evolution [37]. Fig. 9 presents three different buffer evolution trends. In this example, a buffer of size B is divided into two equal parts, B_1 and B_2, and a line marked T that divides them represents a threshold mark. Fig. 9(a) presents a scenario when the buffer refill rate is increasing rapidly, and the buffer occupancy is greater than the threshold point. This implies that the



(a) Buffer level above the threshold and growing



(b) Buffer level below the threshold and growing



(c) Buffer level at the threshold

Fig. 9. Playback Buffer Evolution Trends.

video rate that is being downloaded is conservative, the ABR algorithm may decide to switch to a higher quality level. In Fig. 9(b), the buffer level is below the threshold, and the buffer is growing slowly, which means that the ABR algorithm is too aggressive with its video rate. Thus, it needs to switch down the quality level of its chunk except if the current level is the minimum representation in which case there is nothing an ABR can do. However, when the rate of the buffer occupancy evolution is neutral (as can be seen in Fig. 9(c)) an ABR scheme may opt to maintain its current representation.

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 III-A) is at best imprecise. Upon realising that solely relying on TCP throughput only results in an ABR that is unstable [59], unnecessarily rebuffering [78], requesting sub-optimal video rates [42], and unfair [78] researchers started prioritising buffer occupancy when making adaptation decision. The work presented in [37] is an important milestone in shifting towards buffer-based ABR, it uses the buffer occupancy as the main factor with TCP throughput as an adjustment factor. The algorithm the paper proposes relies on three buffer related properties, i.e., (i) buffer size adjustment factor, (ii) buffer trend adjustment factor, and (ii) video chunk size adjustment. Other research that prioritised buffer occupancy over TCP throughput are [100] and [101].

To the best 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.* [38]. The authors argue that the ultimate purpose of any rate adaptation algorithms is controlling a playback buffer. And the most important task for any ABR is the prevention of rebuffing. Furthermore, buffer occupancy contains 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?' [38].

The algorithm proposed in [38] works as follows: (1) provided the start-up period is passed, the current chunk quality level is increased to the next level if the rate suggested by the rate map exceeds the next higher available video quality level; (2) the current representation is reduced to the next lower quality if the video rate suggested by rate map is lower than the next available quality level lower than the current level, (3) otherwise the current video rate is maintained. Furthermore, the authors of the paper suggest that a segment of the buffer is reserved for the start-up period: the called a "reservoir". While filling the reservoir only chunks with the lowest video rate are requested. They analytically show that the algorithm will never rebuffer provided that the network capacity is greater than the minimum video rate, and will always converge at the video rate that matches the network capacity.

The above buffer-based ABR is initially analytically evaluated. In their subsequent work [75], a large scale experiment is conducted a with millions of randomly selected users from the Netflix streaming service. The result of the experiment was encouraging, even a simple buffer-based algorithm like the one proposed at [38] was able to reduce the number of rebuffering events by 20% compared to the one of the best commercial offering. However, the algorithm's performance with regard to average video quality and the stability of the switching logic could be improved. Thus, they propose a number of additional of improvements, such as the dynamic selection of the reservoir size (based on current chunk size), a rate map based on the chunk size not video rate to account for the VBR nature of chunks and the extension of the reservoir to help improve the algorithm's reliance against temporary network outages.

C. Power-Based ABRs

Delivering high quality video content to a mobile wireless device is a challenging issue because of the limited resources of the access technologies. Furthermore, mobile terminals are usually energy constrained and video streaming is a power intensive exercise. Various researchers have found a significant increase in energy usage when streaming video [102]–[104]. In fact, video streaming can consume as much as twice the energy of playing the same content offline [102]. Understandably, this shortens the battery life and consequently increases the probability of the battery running low in the middle of a streaming session [102].

In general, power consumption is known to be application dependent [130], [131]. Several adaptation techniques have been used to match the application performance to energy requirements. A detailed survey of the various approaches that use content adaptation to manage the energy consumption of the wireless mobile devices is presented in [105]. Inspired by the context sensitivity of energy consumption, researchers have investigated the relationship between video quality and battery energy depletion rate. In [132], the impact of streaming different video formats and spatial resolutions, stored on a server, is investigated. The authors found that low fidelity video, in most cases, consumes less energy. Recently, it was found in [106] that the power consumption of mobile devices heavily depends on the video rate of the content being streamed. For instance, they discovered that changing the video resolution from 1080p to 360p reduces the power consumption by half. It was also observed in [102] that decreasing the video quality level can significantly reduce energy expenditure.

Non-HAS streaming services have attempted to prolong battery life by taking advantage of the differential energy consumption of the video. This is mainly done through dynamically balancing energy conservation and video quality [133]–[135]. Recently, HAS research has started to look at using power as a factor for rate selection [51], [86], [107], [108]. A typical power-based ABR either employs techniques that are known to save energy or use the battery level for rate selection.

Many factors are known to increase power consumption. To save energy, most network interfaces periodically go inactive when not transmitting or receiving data. It is worth noting that the inactivity timer is technology specific. When using HTTP/TCP for video streaming, usually, a persistent connection is established. Hence, even if not downloading any data, the wireless radio remains active, which results in the inactivity period being too short for the interface to enter the sleep mode. In an attempt to reduce energy consumption various ABRs schedule requests in such way that the sleep modes are not unnecessarily reduced, or worse done away with. In [108], a scheme that makes the length of the sleep mode context dependent is proposed. The paper adaptively sets the inactivity timer base on transmission constraints so that the energy saving is maximised. In [109], it has been observed that since users tend to watch about 60% of streaming video to the end [136], pre-fetching may result in energy wastage. Hence, they proposed a scheme that minimises such wastage.

Li *et al.* [51] optimise power consumption by minimising unnecessary 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 [86], a bundled chunk download strategy is presented. Put simply, a client downloads a set of chunks (as bundle), and then waits until its buffer is depleted to a certain threshold before requesting another bundle. Kim *et al.* [110] propose a similar approach where a client pre-fetches a large amount of video content then enters an OFF period until the buffer shrinks to a predefined threshold.

Hosseini *et al.* [111] use an energy specification for each chunk stored at the server. However, this requires an additional attribute to the MPD. The energy specification is derived from the chunk size, which usually depends on the quality level together with the decoding and the rendering complexity of the chunk. The proposed framework maps a client's energy estimation to the right energy specification for each representation. In [112], an energy-aware ABR adapts video quality based on the remaining battery capacity of the mobile client thereby prolonging the battery life.

TABLE III
TYPES OF SCHEDULING USE IN ABR

Types of Scheduling	Research Work
Sequential	[19], [28], [36], [37], [38], [49], [51], [74], [109], [84], [96], [98], [138], [139], [40], [140], [141], [142].
Parallel	[98], [138],, [143], [144], [145]

IV. CHUNK REQUEST SCHEDULING FUNCTION

Chunk scheduling is the process of deciding when to dispatch a chunk request. Generally, a scheduler takes as input a set of parameters, such as the buffer size, target buffer level,[9] and the time the previous download ends. Its output is the time to dispatch the next request to the server.

Scheduling of a chunk requests can either be ***sequential*** or ***parallel*** [71]. A sequential scheduler request 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 subsegment. A parallel scheduler is mainly used when a client intends to use multiple interfaces and/or wants to access content from multiple locations. Table III presents a summary of the research that employs either of the discussed scheduling methods.

A. Sequential Schedule

The most basic of all sequential scheduling techniques used in HAS services is called ***progressive dispatch***. In the progressive dispatch, a request is made as soon as a response is received, this aggressively ramps up a buffer. Incidentally, if allowed to continue unabated progressive dispatch can easily overwhelm a client that has limited playback buffer. During the start-up period or when the buffer level is below a predefined 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 [36] and YouTube [74], as well as some non-commercial players (e.g., DAVVI [137], and FESTIVE [49] are known to use progressive dispatch to fill up their buffer. Furthermore, it is argued in [40] that for a client to get its fair share of the available bandwidth, progressive dispatch should always be used before the download of the maximum available video quality level.

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 one 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 [38], or to save energy when streaming

[9]Note this depends on the implementation.

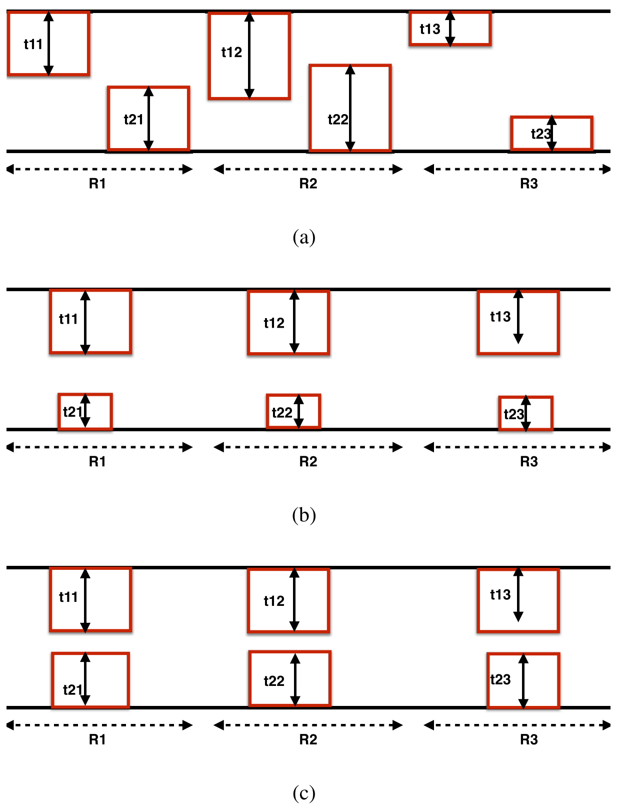


Fig. 10. Unstable, Unfair and underutilised scenario.

in mobile environments [51], [138]. This effectively results in an ON and OFF traffic pattern. Most implementations derive the time interval between requests (i.e., the duration of the OFF period) from the chunk size [36], [74].

In [79], the performance of periodic dispatch is investigated. The paper found that the use of periodic dispatch can cause a deterioration in TCP throughput. The authors then suggest the use of a long video segment as a remedy. While [42] look at the behaviour of periodic dispatch scheduling when multiple players are trying to share a bottleneck link and observe that there are issues with instability, unfairness, and underutilisation of resources. The examples shown in Fig. 10 illustrate the problems. As can be seen in Fig. 10(a), at R1, the two competing players have non-overlapping downloads, as a result of which each player overestimated the available bandwidth. Therefore, when making the subsequent request (shown through R2) both players request a higher video rate than the network can sustain ($t12 + t22 > T$). This situation results in congestion, which forces both players to reduce the quality level of their next request. However, in the subsequent round (R3) the sum of the bitrates of the requested chunks is less the capacity of the network, i.e, ($t12 + t22 < T$). They found this cycle repeating, hence causing video rate instability.

Another scenario is when the ON period of one player lies within the ON period of the of another player as can be seen in Fig. 10(b). This normally happens when players are requesting unequal video rates and the two players have a false estimate of the available bandwidth. Consequently, the player downloading the higher rate estimates more than the one downloading lower video rate. Therefore, the players remain in a stable but unfair equilibrium. The third problem manifests itself when the players' downloads are perfectly aligned, but both are requesting a low video rate (see Fig. 10(c)). In this case, they reach a stable and fair estimate but with video rates less than their fair share of the available bandwidth, i.e., $t11 + t21 < T$, $t12 + t22 < T$. This situation results in the underutilisation of the network. The cause of these problems observed by Jiang *et al.* [49] is synchronised scheduling.

In [96], three likely solutions to both the unfairness and the oscillatory nature of the above switching logic is identified. The first approach is to randomise chunks inter-arrival time, and secondly, a back-off period is introduced. Whenever a client switches down the quality level of its download, it should not increase its OFF period for some time. Finally, rather than using a fixed range within which the inter-arrival time of chunks is determined, a threshold should be set, which will determine when a system should operate aggressively or otherwise. The authors tested the first approach and found through simulation an improvement in the players' fairness. In their later work [139], they investigated a scenario whereby each of the competing sessions is assigned a fixed but unique inter-arrival time. They report a similar performance as in the case of the randomisation of the inter-arrival time. However, applying this in a practical system will require having the video stream available in multiple segment sizes. In [49], the value of the target buffer is randomised. The paper shows that by doing this the scheduling is still periodic but individual periods are independent of the player's start time and the issues reported by [42] are mitigated.

For any video streaming application, avoiding rebuffering and streaming the best possible video quality are the two main objectives of any rate adaptation algorithm [144]. However, live streaming sessions have an additional requirement, i.e., *liveliness*. The concept of liveness means late chunks are skipped if the stream lags too far behind its deadline. It is noteworthy that for live streaming the buffer is small by necessity in order to avoid long delays. Consequently, live streaming has stricter deadlines [98]. Additionally, scheduling in live streaming also has to take into consideration the fact that a client's arrival time decides at which point its session begins. So far none of the discussed scheduling algorithms considers these additional requirements of live streaming. While periodic dispatch trades timeliness for efficiency in buffer management, progressive dispatch is a greedy algorithm that may request chunks not yet available. Hence, an algorithm is required that can cope with 'deadline-misses' and has a notion of 'first request'. Kupka *et al.* [140] evaluate a series of segment streaming strategies, with each strategy being a possible outcome of a set of the combinations of the following four options (each option has two values so there are 16 possible combinations in the set):

- First request: a client may request the most recent chunk or wait for next available chunk.
- Play-out start time: a client may start play-out immediately or delay the play-out.

- Next request: a client may request a chunk before download finishes or before the play-out finishes.
- Deadline miss handling: a client may start playing from the beginning of the download or skip a portion equal to the deadline-miss from the download.

The result of extensive experiments [140] shows that the best combination is when a client requests the most recent chunk in its first request, starts play-out immediately, dispatches requests before play-out finishes and start playing each chunk downloaded from the beginning. This combination is found to avoid the synchronisation of client requests resulting in a high-quality request and short delays.

A different approach is taken in [68], [69], and [145], 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 [68], [146] 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 [68].

B. Parallel Schedule

Parallel scheduling is required when a client has multiple network interfaces and uses some or all the interfaces simultaneously for a 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 [98]. In [141], the case where HTTP range retrieval is used to sequentially request chunks by a multi-homed client is studied. The performance of the technique is found to be dependent on segment size. The authors found that using small segments increases the overhead, which results in a reduction of aggregate throughput. A large segment size is found to significantly increase both the start-up delay and the buffer requirements. 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 [142], they investigate parallelising the schedule since finding an optimal segment size, they argued, imposes a trade-off between the throughput and the start-up latency. They proposed a technique based on an HTTP pipeline that allows a client to send a request without the need to wait for a response. The technique starts by interleaving byte range requests, thereafter each interface may send a request immediately after receiving a response. The reason for interleaving at the start-up is to prevent interfaces from sending requests in sync. By ensuring that the server is always busy processing and responding to requests the efficiency of throughput aggregation is found to be close to the optimal level. However, the work was done in the context of progressive download. In [97], a similar approach but within a HAS by extending DAVVI [137] with multi-homing capability [137] is used. Experimental results show that the scheduler reduces video interruption and improves average video quality, even when using heterogeneous multiple wireless interfaces.

In contrast to the approaches that use the HTTP range retrieval [143] proposes 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 [71], [143]). 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 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 that the set upper bound.

V. 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*). The adaptation logic usually takes information regarding the available resources, the schedule of the next chunk, and the set of all the possible representations, as its inputs. Then returns a representation of a chunk to be downloaded. However, when a server presents to a client some 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 basic 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 (even though it is simple to implement). Because it is very sensitive to the time-varying nature of the resources that ABR schemes commonly rely on, which can make its outcome oscillatory, abrupt [42], [74], and unfair in allocating bandwidth to competing clients [78].

Traditionally, most HAS players rely on an adaptation logic that mimics the concept of an *additive increase multiplicative decrease* (AIMD) control scheme. This is because AIMD easily converges to an efficient and fair state regardless of the starting point [14]. It helps in reducing the fluctuation of video quality and abrupt change in video quality that are

TABLE IV
DIFFERENT TYPES OF METHOD USE FOR ADAPTATION MODULE

Types of Scheduling	**Research Work**
Heuristic Based	[19], [28], [46], [49], [77], [150].
Control Theory	[30], [37], [100], [151], [152], [153], [154].
Optimisation Techniques	[50], [87], [155], [156], [157].
Artificial Intelligence Techniques	[158], [159], [160], [161], [162], [163], [164], [165]

not appreciated by users [46]. Further, it was demonstrated in [83], [128], and [147] that when the video quality is high an aggressive increase in the current rate does not necessarily translate into an improvement in the user-perceived quality. With AIMD-like approaches, a client increases the video rate of a chunk request in a stepwise manner after each request-response transaction. When the system resources dwindle, the video rate is aggressively reduced to avoid buffer starvation. As discussed in Section IV, the download of low video rate chunks allows for a rapid refill of a buffer. However, it has been shown in [80] that the simpler the adaptation logic, the better, regardless of the approach used in building an adaptation module. Table IV presents the summary of some of the techniques commonly used in implementing and adaptation logic.

A. Heuristic Based Adaptation

Most of the early ABR schemes are based on heuristics. For instance, Liu *et al.* [19] heuristically implemented an AIMD-like adaptation logic that employs a stepwise switch-up and aggressive switch-down logic. They argue that this technique prevents video artefacts that might happen if the switch-up is more aggressive, and with an aggressive switch-down rate of buffer refill becomes faster. Thus, preventing video interruption. However, the algorithm was found to intermittently chose suboptimal representation and is unstable as it oscillates between different video quality levels.

Another player with a heuristic based adaptation module is FESTIVE [49]. The player 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. The delayed update is a score that measures a trade-off between efficiency/fairness and stability, which allows a player to improve its stability. In [46], the effect of video quality transition on QoE, is studied. The authors report that a sudden drop in 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 suboptimal choice, by improving stability they are able to enhance the subjective user experience. Experiments in [36] 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 [28], AdapTech Streaming [77], and the Akamai HD Video Streaming services [148].

B. Control Theory Based Adaptation

It is difficult to use heuristics to design an algorithm that is predictable and mathematically describable. Thus, recently there has been many 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 [164]. The controller is the part of the control system that manipulates input to produce the desired output. A typical controller computes the difference between a measured variable and a target point as a process error. The goal is to reduce this error by adjusting the process input parameters.

De Cicco *et al.* [149] propose an adaptation logic based on feedback control. The video rate 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 that 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. Experiments confirm that the controller selects the highest video rate that the available bandwidth can maintain, though the measurement is done server side. In [37], a control theoretic client side rate adaptation that makes a trade-off between the stability in video rate and bandwidth utilisation has been proposed. In [152], 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 decision on which bitrate is the most optimal in maximising QoE. Other papers that propose adaptation functions that are implemented using control theory are [30], [100], [150], and [151].

C. Optimisation Based Adaptation

In [50], a different approach has been tried. The paper uses an optimisation technique for rate adaptation algorithm (called Intelligent Bitrate Switching based Adaptive Video Streaming). And modelled the adaptation logic as an optimisation problem, which maximises benefit and minimises penalty. The quality level of a chunk represents the benefit, with higher video rate 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 constraint of a minimum number of rebuffers.

Another adaptation logic based on optimisation techniques is presented in [87].

At the network provider side, there is a need to arbitrate the allocation of network resource among competing clients. Hence, according to [153] and [154], the support for coordinated management and global optimisation is imperative. 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 subscription contract. Joseph and de Veciana [155] 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.

D. Artificial Intelligence Based Adaptation

In [156], it is argued that since a change in video quality 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*. The system has three parts, a sensor, a controller, and an actuator. The sensor is equivalent to a resource estimation module. The controller is the part that is responsible for the adaptation, while the actuator realises the controller decision. The fuzzy controller is in turn made up of three components, fuzzifier, fuzzy interface engine, and defuzzifier. The fuzzifier takes input (in this case is the estimated throughput) and converts it into a format that the controller understands. The fuzzy interface engine takes the fuzzified input and produces an output based on rules generated from the domain knowledge and expert experience. The defuzzifier converts the output produced by the fuzzy interface engine to a form other parts of the system will understand, i.e., the video rate of the chunk to be downloaded (for more detail on how each part works see [156]). They experimentally found the technique to be responsive to changes in network conditions. However, the results show unnecessary instability even in the presence of stable instant throughput.

In [157], fuzzy logic is employed to adapt the video rate to the changing network conditions, but unlike [156] 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 Sobhani *et al.* [158] propose an AIMD-like fuzzy controller that considers 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 difficult to define a set of linguistic rules based on it [165]. Another artificial intelligence (AI) technique used in the video rate adaptation that requires no expert input is machine learning (ML) [159]–[162]. With ML techniques, a client can learn to adapt its video quality to the changing context without the need for any human intervention. In [161] a decision tree based random forest classification (i.e., a composite classification algorithm) is used to map network related features onto the video rate. Rather than proposing a new algorithm they opted to use the scheme in improving the accuracy of existing adaptation algorithms. In summary, the scheme trained the classification model using a dataset provided in [166]. The classifier is then used to predict the current request or any future video request. The training can be done either on-line or off-line. Simulation results show an improvement in the prediction accuracy of the baseline algorithms.

Classification schemes generally require a training dataset. However, in 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 specific context based on a feedback from its environment. To do this an adaptation module interacts with its environment by sensing factors that are expected to influence it decision. For example, van der Hooft *et al.* [163] use the average and the mean absolute difference in bandwidth, while [159] and [160] use both the information about buffer state changes and available bandwidth. Then the agent acts typically by changing the video rate to incrementally maximise its reward (such as improving the Mean Opinion score and reducing the rebuffering) [163].

VI. Interactions of Components

ABR module, like most complex systems, is composed of a number of components, which can be grouped into three elements. Furthermore, it can be observed that these components are mostly treated independently from one another. This allows us to treat them in greater detail. However, as can be seen from Fig. 11 these components of ABR exhibit non-trivial interactions.

Generally, the relationship and interaction between throughput estimation subsystem and buffer management function receive the highest attention [36], [74], [78]. It has been found that the more the available network capacity, the faster the chunk download [37], therefore, the faster the replenishment rate of the buffer. However, many algorithms such as [28] and [59], try to maintain their buffer at a predefined level. For this, the scheduling function activates the periodic dispatch, which results in an ON-OFF traffic pattern. As discussed in Section III-A4, this traffic pattern results in the drop of TCP throughput, which in turn reduces the speed of the buffer replenishment. Obviously, 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 [38], [40], and [75] it is recommended that the periodic dispatch is only activated when a buffer is full or nearly full. In other words, while ABR scheme is ramping-up its video rate only the progressive dispatch is used. They argue that this will ensure that provided the highest video rate

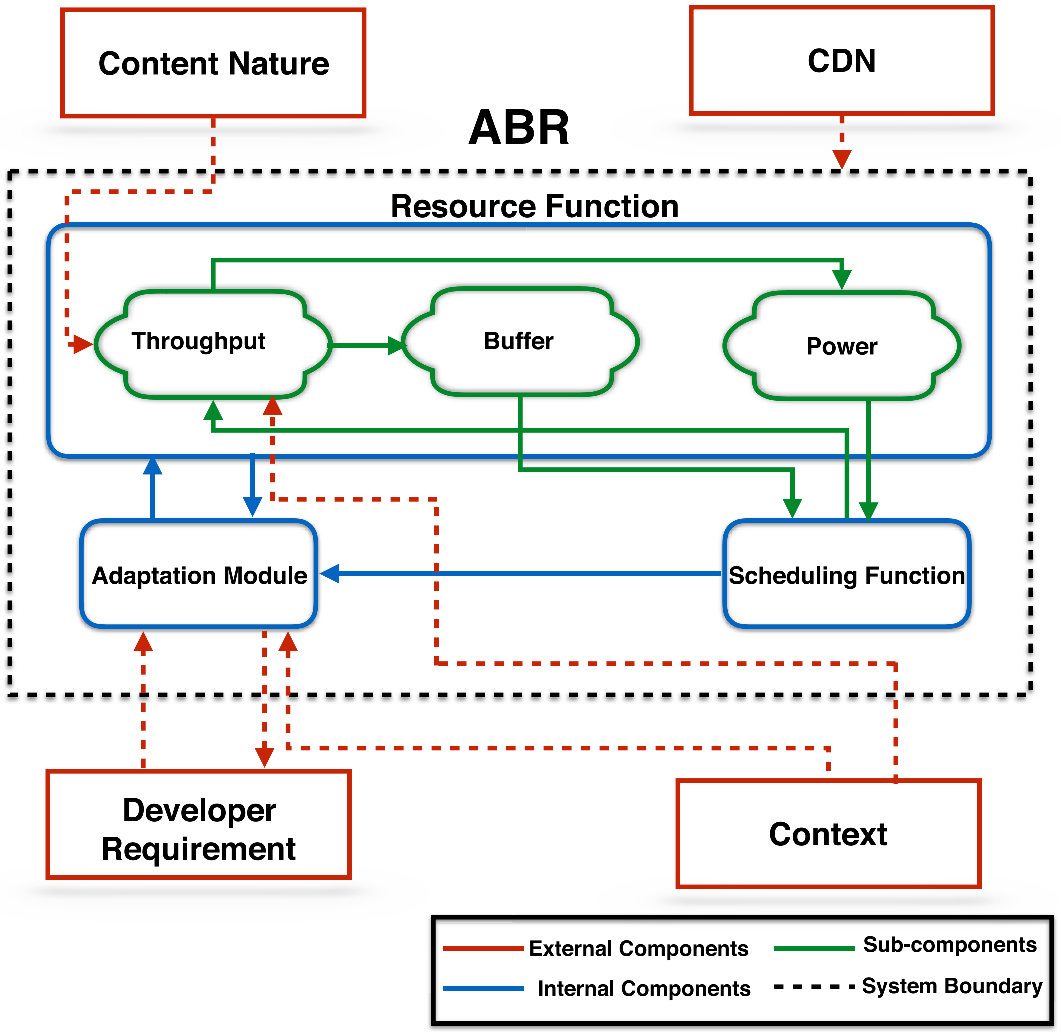


Fig. 11. Interaction of ABR Components.

is not reached the OFF period is not activated. However, this may not completely solve the problem since a back-to-back strategy introduces at least one RTT delay between contiguous chunk requests, which understandably has an impact on the TCP throughput.[10] A better strategy will be to send the next chuck request to the server, perhaps through the uplink, before the current chunk download finishes.

Another cyclic relationship is between power consumption and the throughput. As outlined in Section III-C, power consumption is proportional to the network utilisation. Furthermore, to control power consumption [102]–[104], devices usually employ the inactivity period, which may or may not correspond to the off-period of the periodic dispatch scheduling scheme. One solution to this trade-off as proposed in [167], is the employ of a mechanism that optimises video chunk size in order to minimise the energy of wireless interface. The authors' reason that since bigger chunk (i.e., with longer duration) allows the throughput estimation module to better estimate the available capacity , while at the same time saving tail energy.

However, these two subcomponents (buffer and power management units) of the resource estimation function are not the only subsystems interacting with throughput estimation module, others are: content nature, CDN, and the streaming context. 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, or where CDN providers may locate their data centres.

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 designers constraints and policies. For example, video oscillation should not exceed a certain threshold [37], or the start-up delay must be within a particular range [29], [77]. Therefore, the adaptation logic must ensure that the selected video rate does not result in the violation of any of the constraints. It should be noted that some of these constraints may evolve with time. For instance, in [29] and [76] the adaptation

[10]This is because of the well-known fact that TCP throughput is inversely proportion to RTT.

TABLE V
IMPACT OF CHUNK SIZE ON THE BEHAVIOR OF ABR

Effect on ABR	**Large Chunk Size**	**Small Chunk Size**
Rebuffering	More rebuffers [170].	Less rebuffers [170].
Encoding Efficiency	Low overhead [171].	High overhead [171].
Adaptability	Low adaptability[170].	High adaptability [170].
Quality Fluctuation	Reduce fluctuation [84].	Increase fluctuation [84].
Bandwidth Utilisation	Improves utilisation [79].	reduce utilisation [79].
Start-up Delay	Long delay [172].	Short delay [172].
Fairness	Improve fairness [78].	Reduce fairness [78].

module most evolves the output video rate in a logistic manner, and in [38] and [75] it has to start with lowest video rate and maintains it until the 'reservoir is filled' (see Section III-B for detail). In this sort of situation, the current action of the adaptation module decides the next thing to do; 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 buffer replenishment rate and the video rate of the requested chunk since chuck with lower video rate contains less data and vice versa. Hence, assuming a constant available bandwidth, the buffer replenishment rate has an inverse relationship with video rate. However, the throughput perceived has a proportional relationship with the video rate. As discussed in Section III-A, there is a feedback loop between the data downloaded and the TCP mechanism, with longer download allowing TCP to have a better chance of reaching a steady state. The next section will talk more on the impact of chunk type on the throughput perceived by the estimation module.

VII. IMPACT OF CONTENT PREPARATION

One basic requirement of HAS is that a video file has to be segmented, with each chunk being independently decodable. Equally important is that multiple encoded versions of each of the chunks are to be provided at the server [32], [47]. Thus, determining the optimal size of the segments and appropriate set of representations becomes a critical challenge. Furthermore, since any technique that guarantees fine granularity, at least in theory, can be used to provide video file segmentation, it is interesting to look at how other encoding techniques affect the effectiveness of ABR scheme. In this paper, we concentrate on multi-layered coding, i.e., Scalable Video Coding (SVC).

A. Chunk Characteristics

Various segment sizes have been used by a number of HAS implementations reported in the literature, e.g., Apple HLS uses 10s [47], Adobe HDS uses 2-5s [170], MSS uses 2s [33], and 1s is used in [68]. In [171], a dataset is provided with segment sizes of 2, 4, 6, 10 and 15s. Other datasets that provide video with multiple segments sizes are the YouTube Mpeg-Dash dataset [172] and 4Ever project ultra-high definition HEVC DASH dataset [173]. In [174] a dataset is provided in both H.264 and H.265 with encoding rates similar to those provided by commercial content distribution providers such as Netflix and YouTube. BITMOVIN maintains a list of the free and public MPEG-DASH test streams and datasets [175].

Several performance evaluation studies have been conducted to ascertain the impact of chunk size on the behaviour of ABR [79], [168], [171]. In [79], it was found that the larger the segment size, the higher the network resource utilisation. This is because larger chunks take longer to download, and hence are more likely to ensure that TCP reaches the steady state. Similarly, it was shown in [84] that small chunk sizes result in an inaccurate estimation of the available bandwidth. However, Yao *et al.* [168] noted that the use of large chunks sizes comes at the expense of the adaptability of ABR algorithm when the bandwidth rapidly fluctuates. Once a chunk is in transit and the system condition changes, a client typically has only two options, i.e., either to continue and suffer suboptimal performance or to abort the download and waste valuable bandwidth. Larger chunk sizes are also found to increase the end-to-end delay in live streaming [66]. In contrast, small chunk sizes increase encoding overhead [169], video quality fluctuation [84], and unfairness among competing clients [78]. Table V presents a summary of the impact of chunk size on the performance of ABR. By employing server push technique [69], the authors of the paper are able to reduce the number of request-response associated with small chunk size, which allows them to use "super-short" chunks with sub-second duration.

The selection of appropriate video chunk size has to consider several competing factors. Lederer *et al.* [171] attempt to find the optimal video chunk for a network setting. They found it to be dependent on the nature of the connection established (i.e., either persistent or not). Instead of finding the optimal segment size Liu *et al.* [84], Lievens *et al.* [169], and Jeong and Chung [176] attempt a different approach by opting for variable chunk sizes. In [169] chunks with different sizes are provided, with low quality chunks having small segment sizes and high video rate chunks having large sizes. Liu *et al.* [84] monitor the network and then request a chunk size based on the variance of the TCP congestion window. Similarly, in [176] the segment duration determination algorithm called S-DASH is proposed. The scheme determines the chunk size to be downloaded based on the TCP throughput variation.

Usually, a client is presented with a set of chunks of different quality levels from which to choose from. Two things become obviously important (i) the cardinality of the set of the available video representations (e.g., in [171] and [173] a number of datasets have been proposed with varying cardinality), and (ii) the difference in bitrate between successive profiles. For example, in [87] equally spaced profiles have been used. The high cardinality of a set of video rates allows a provider to cater for more contexts but at the cost of higher storage overhead. From an ABR perspective, a large set of representations means higher switching rates at the benefit of improved adaptability [81]. Thang *et al.* [165] have shown that different sets of representations have varying effects on the behaviour of ABR. For example, equally spaced chunk profiles are found to decrease the user perceived experience by increasing the variation in the distortion difference between contiguous video quality switching. As a solution to this, they propose a set of representations whose elements are spaced based on just noticeable differences (JND). They found that this can achieve an improved QoE and buffer stability. Correspondingly, Lederer *et al.* [171] rely on PSNR based heuristic to derive the distance between representations. Toni *et al.* [64] presented criteria and practical guidelines of finding an optimal set of representations that maximises QoE. However, it was found in [81], through simulation, that the set of representations provided by most of the commercial content providers, e.g., Netflix and YouTube and the one proposed in [64] do not provide enough flexibility for heterogeneous clients when a network is overloaded.

B. Layered Coding in HAS

So far, we have assumed that every chunk is self-contained and independently encoded. To some extent, this is a valid assumption since H.264/AVC is the most widely used video encoding technique. However, having each chunk self-contained has its downsides. For each representation, all chunks have to be encoded and stored separately. For instance, a 1hr 30min video file is segmented into 2s chunks, with five (5) different quality levels will result in 5400 x 5 chunks. According to [177] the MSS service incurs between 200% to 300% storage overhead compared to having only the highest quality level available. More so, they claim that if intermediate systems such as a CDN is employed, to get a similar cache hit-ratio as in the case of a single representation additional 3 to 4 times storage space is needed. de la Fuente *et al.* [178] also confirm the suboptimal performance of chunk-based HAS in terms of caching efficiency. Furthermore, with the storage overhead comes the need for additional bandwidth to transport the additional chunks [179].

The essence of an ABR is to enable clients to adapt the video quality to different or changing contextual conditions. In chunk-based HAS services, a segment must be delivered in its entirety or not. Certainly, this will affect the adaptability of an ABR logic. One way of solving this dilemma is to employ small chunks. However, it was found that [84] this inaccurately estimate the available bandwidth, which results in a fluctuating rate adaptation since small chunks do not allow TCP to reach a steady state. A better solution argued [178] is to use the layered coding. Scalable Video Coding (SVC) [180] is an extension of H.264/AVC for layered video coding. Layered coding allows the encoding of a video into a number of layers, composed of a base layer (representing the least quality level) and a number of enhancement layers, with each enhancement layer improving the viewing quality. With SVC, a video is encoded once and decoded based on a frame rate, resolution, and/or fidelity requirements. It enables the encoder to remove a 'part of the video bit stream in order to adapt it to the various needs or preferences of the end users as well as the varying terminal capabilities or network conditions' [180]. SVC allows for three different kinds of scalability, i.e., temporal, spatial and quality. With temporal scalability, the base layer represents the source content with a reduced frame rate, while in the case of spatial scalability the resolution is reduced. The quality scalability presents a scenario where the base layer has the least fidelity (mostly measured in signal-to-noise ratio). Any addition of the enhancement layer to a base layer increases the frame rate, the resolution or the fidelity as the case may be. Nevertheless, any combination of the three approaches is equally possible. A detail discussion on SVC can be found in [180]–[182].

Intuitively, with encode once and decode multiple times the issue of storage overhead can be mitigated. HAS is designed to be encoding technology agnostic. It should not matter whether AVC is used or not. However, it is worth noting that every technology has specific characteristics. Therefore, researchers always attempt to exploit such differences to enhance a system. The initial discussion has concentrated on implementations that incorporate SVC into some of the existing ABR schemes without taking into account the peculiarity of its multi-layered nature.

Even with layered coding, a video file needs to be chopped into segments to better suit HAS. There are various chunk creation strategies proposed in the literature. In [183], a video file is segmented into self-decodable units. Each unit is the equivalent to a chunk, which is made-up of blocks with each block representing a layer. Xiang *et al.* [184] take a different approach, they segment the encoded video along the layers, and then chop each layer into segments. Therefore, a call for a chunk implies a request for a particular layer. Grafl *et al.* [185] use multiple independent groups of chunks, with each group of chunks having the same class of base layer such that they represent a particular resolution, and layers within a chunk are used for quality adaptation. They called the approach a hybrid SVC-HAS.

Theoretically, SVC video chunks can easily substitute AVC chunks in any of the plethoras of the traditional rate adaptive schemes discussed. Famaey *et al.* [186] use SVC chunks with the open source version of the MSS rate adaptation algorithm [73]. The quality selection is based on selecting a base

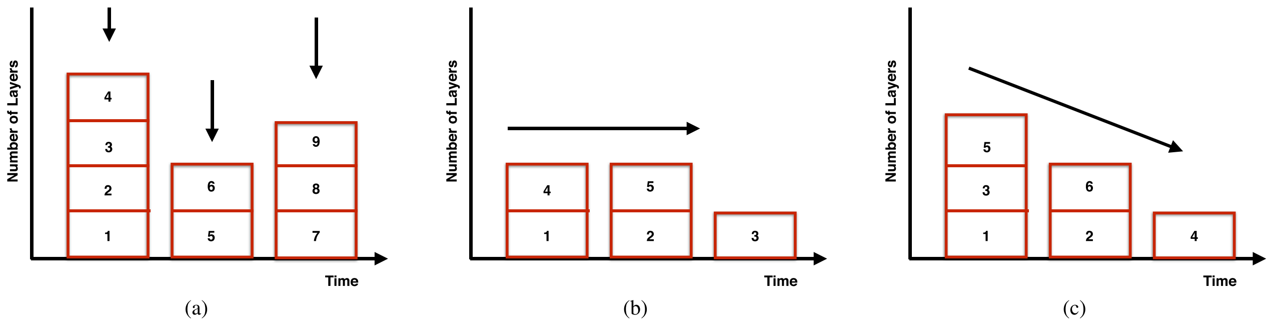


Fig. 12. Illustration of How SVC layers are downloaded in HAS.

layer and the required enhancement layers. For example, an ABR decides to request the third quality level, the base layer and the subsequent two enhancement layers are downloaded. Müller *et al.* [187] and Sieber *et al.* [188] successfully adapted SVC chunks to the ABR scheme proposed in [41], which is originally designed for a single layered AVC.

The techniques discussed so far are based on vertical adaptation. In the vertical adaptation, a base layer and the entire required enhancement layers are downloaded within a time slot allocated to a chunk. As can be seen in Fig. 12(a), only after all the required enhancement layers are downloaded will a new base layer be requested. The vertical adaptation is mostly employed when the intention is to maximise the video quality. Sieber *et al.* [188] propose a horizontal adaptation, i.e., a fixed number of base layers are first buffered, (see Fig. 12(b)) and then in subsequent rounds the enhancement layers are downloaded (with each round requesting only a particular enhancement layer). The logic behind this is to ensure consistent quality viewing.

Both the vertical and the horizontal adaptation schemes do not take advantage of the enhanced features of SVC. For the duration of each chunk, while using SVC, multiple requests are made. In the case of the vertical heuristics, the base layer and the individual enhancement layers are separately downloaded, while in the horizontal schemes the contiguous layers are individually downloaded. When the latter feature is exploited an improved granularity of rate adaptation is obtained, and the rate adaptation module can abort download midway without much penalty.

To get the best of both worlds, Andelin *et al.* [189] propose a diagonal heuristic. The rate adaptation decisions are now made after each layer is downloaded. In the diagonal scheme, a client can either download enhancement layers of the current chunk (therefore increasing the current quality) or download the next segment and ensure better quality in future. The heuristic uses a slope to determine how a client alternates between backfilling and pre-fetching (see Figure 12(c)). When backfilling, the quality of the current base layer is enhanced, while pre-fetching downloads a layer that may be required in the near future. The gradient of the slope is configurable. When pre-fetching is desired the slope is made flatter, and a steeper slope used when the goal is to enhance the current segment.

Many researchers have investigated the performance of SVC in HAS. Amongst the earliest work is [178]. The paper investigated the effect the two encodings schemes have on caching efficiency and found SVC to be more efficient due to the fact that the base layer is always available. Another advantage of the use of SVC is an improved responsiveness to the change in network conditions [177]. Because SVC allows for finer granularity since it works at layer boundary, this enables a client to abort a download without much overhead.

However, layered downloads substantially increase the number of request-response transactions. This is because after each request-response cycle there is a waiting period of at least one RTT. Consequently, SVC based ABRs are more vulnerable to higher RTTs. This has been confirmed by both [177], [186]. It should be noted that RTT is inversely proportional to throughput. This means SVC will perform badly in low throughput conditions.

We have earlier seen that SVC-based HAS services can substantially reduce storage requirements. Nevertheless, for each addition of an enhancement layer in SVC video, there is at least 10% encoding overhead [180]. This results in the demand for more bandwidth by SVC-based services. Kalva *et al.* [190] have compared the financial cost of the storage reduction to the cost of the increase in bandwidth requirement by SVC-based services, and found that the latter outweighs the former.

VIII. Context Management

Until recently, video quality adaptation decisions have been mostly based on the system level factors discussed in Section III. 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 [191]–[195]. Context is defined as ‘any information that assists in determining a situation(s) related to a user, network or device’ [191]. As observed in [192] 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, the more precise its definition is. However, this will be difficult to manage and model. Therefore, to better manage this complexity Mitra *et al.* [191] grouped these parameters into four dimensions:

1) Device related parameters, e.g., screen size and 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, and RTT.

A context vector, $CV = \{1 \ldots n\}$, is a set of combination of parameters from the above groups, which uniquely identify a context. When the *CV* consists of parameters from one group, for example, 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 III 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.

A. 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 III, 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 [192] and [196], 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 [91], [197]–[200].

In [198], a location-based bandwidth lookup service is proposed to help streaming clients better predict the available bandwidth. 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 [200], 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 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 [91], 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, which 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 [201] 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 time of the day. Relying on the geostatistical methods, the authors analyse the impact of both space and time on bandwidth prediction. They then used the spatio-temporal 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.

B. 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 at 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 [199], 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 [192] and [202], location information is used to find the 'close-to-optimal' buffering strategy that prevents video stalling in an area of limited connectivity. Basically, 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 to or the longer is 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

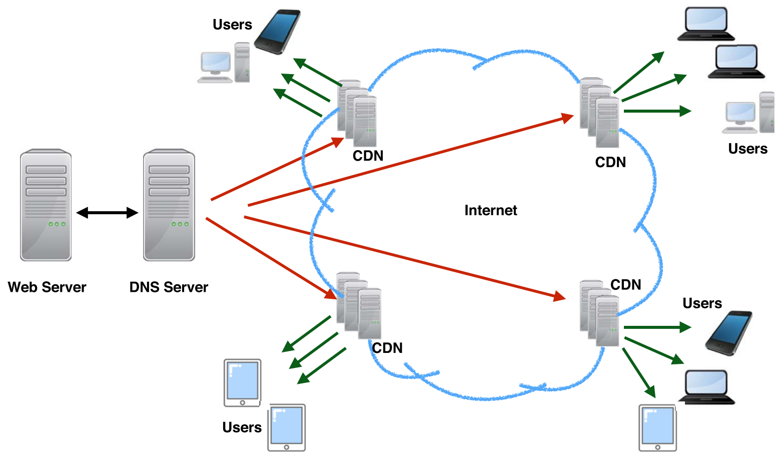


Fig. 13. Typical architecture of a CDN.

that a client has the ability to predict its path and knows where the areas of interest are located.

IX. Content Delivery Network

To increase content availability, scalability, and reduce access latencies video content providers are increasingly relying on CDN providers (e.g., Akamai, Limelight) for video content distribution. A CDN is 'a collaborative collection of network elements spanning the Internet, where content is replicated over mirrored Web servers, located at the edge of Internet services provider's network to which end user is connected' [203]. As can be seen in Fig. 13, in its most basic form, a CDN is composed of a number of *edge servers* that are geographically distributed and redirection nodes, e.g., DNS servers. Content is cached at the edge servers, which are responsible for content delivery to a client. A client first gets the URI of the desired chunk from MPD, and typically sends a GET request for it. The CDN, on receipt of the client request, using its redirection mechanism reroutes it to the most appropriate edge server. The process of edge server selection is a challenging task, since as observed in [203], some nodes may be unavailable, or overloaded. However, to make a redirection decision, a typical CDN relies on the following metrics: (1) proximity which measures the distance between client and edge servers; (20 load, which indicates the status of bandwidth and CPU; (3) and the possibility of content availability at the edge server [71], [204].

Recall from Fig. 11, CDNs do have an impact on the performance of an ABR schemes that request content cached by it. In fact, it has been suggested in [19] and [205] that CDN infrastructure is more likely to be the bottleneck than the last mile connection in services that use CDN. This challenge may manifest as a result of either the content distribution across the edged servers (caching) and/or latencies incurred as a result of request redirection [206].

A. Impact of Caching

Caching is widely used to reduce latencies by bringing content closer to the users. However, since in most situation the available content is more than the capacity of the cache, content acquisition and replacement mechanisms are critical to the effectiveness of a caching system. Content is either dynamically acquired, that is, content is only requested from an origin server when a cache miss occurred; or proactively pushed to the cache by the CDN [207]. In either case, the link between the origin server and edge server can be the new bottleneck. Lee *et al.* [72], Zhou *et al.* [205], Liang *et al.* [208], and Juluri and Medhi [209] investigate the impact of either a congested or slow link between the origin and edge servers. In [72], it was found that when the available bandwidth between a client and the edge server is more than between the edge-origin link, an ABR scheme overestimates the available path bandwidth when requesting a cached video chuck. If the ABR adaptation module uses the throughput estimate derived from downloading a cached chunk to request a chunk that is not available on the edged server, a high latency that manifests as a reduction in throughput is experienced by the client. This happens because the edge server has to get the chunk from the origin server before sending it to the client over a link with a capacity less than the requested video rate. The authors found this scenario keep on reoccurring, hence, causing the video rate oscillation. To prevent this, the authors use a traffic shaping technique to ensure that an ABR scheme does request chunks with video higher than the path capacity.

In [209], a different approach is proposed. The proposed scheme reduces the chances of the cache miss by pre-fetching video chunks. Through using the knowledge of the available path capacity and the adaptation logic, the proposed scheme is able to predict the video rate of likely chunk to be requested next and then pre-fetch it. In [208], a similar approach that requires no knowledge of the ABR running on a video player is proposed. However, in this case, only the information about the capacity of the origin-edge link is used. The result shows that by improving the hit ratio, the impact of the edge-origin link is reduced. Hence, an overall improvement of the video rate is achieved.

Another source of a drop in capacity at a the edge-origin link is the ability of origin server to serve all the requests from the different edge servers since naturally, it has a limited fan-out. Even in more advanced CDNs where the edge server is served by intermediate level servers, which are in turn served by the origin server, the intermediate node fan-out can be overwhelmed. To prevent a scenario whereby the capacity of either the edge or the intermediate nodes is exceeded, Zhou *et al.* [205] propose a support network that ensures that the nodes that receive chunks from the origin server cooperate to boost the capacity of the origin server.

B. Impact of Redirection

The URI obtained by a client after parsing an MPD file usually points to the origin server [207]. When CDN is in the employ the client's request has to be redirected to the appropriate CDN, which then reroutes the request to the selected edge server, typically using DNS mechanism [210]. The edge server selection protocols are outside the scope of this paper, for more detail see [203]. The selected edge server will then begin the process of responding to the client request. The cost of this process observed [206], [207] is an increase in delay, which reduces the throughput perceived by the throughput estimation module.

In [206] and [207], two solutions are proposed to the request redirection problem. Both solutions rely on writing the MDP. In the first proposal, the MDP is rewritten to point at the request router that then reroutes the client request to the right edge server. In the second solution, the MDP is rewritten to point directly at the appropriate edge server, which allows the client to directly get content without any redirection. The result of simulation shows a reduction in the start-up delay and the number of video stalls. Additionally, an increase in the video quality is also observed. However, it is noted in [211] that when using the MDP-rewriting scheme, a node may be overloaded or even be out of reach without the knowledge of the client, such that only after sending a request would it receive the '404' error, which may result in video stall. To prevent this, the authors of the paper proposed a mechanism that allows the CDN to either push a new MDP or trigger a request new MPD event at the client when node become unavailable.

X. Lessons Learned

The HAS service ecosystem is composed of various stakeholders each with its own set of requirements that are often at odds with each other. Hence, while designing an ABR scheme that meets these, possibly divergent requirements, is a non trivial exercise. Various design paradigms, such as network-assisted and server-side schemes, have been proposed. The choice of a design paradigm dictates which stakeholder becomes an active or a passive participant.

However, regardless of the paradigm used, users are mostly passive stakeholders. In other words, they passively consume whatever video quality the system provides them with. Nonetheless, this does not reduce the importance of their concerns. It rather implies that while the HAS service is running a user has little or no control in how an ABR operates. Furthermore, when a client-side design paradigm is employed, as in this paper, a network operator becomes more or less a passive stakeholder too. Even in the case whereby a content provider decides to locate its cache at the ISP location, for example, Netflix currently partners with ISPs to localise it content using Open Connect Appliance [212], this can only help an ISP localise its traffic but does not give any control over the ABR scheme running at the client-side. This fact has resulted in a number ISPs throttling traffic originating from content providers [213]. The other two stakeholders are an ABR algorithm designer and a content provider.

A. Summary of Important Points

In order to build an effective HAS service, ABR designers have to consider the requirements of both the active and the passive stakeholders. The following are some lessons derived from our earlier discussions that can help ABR algorithms designer to build services that satisfy multitude of requirements.

- *Resource estimation:*
  - ⋆ The choice of scheduling policy is as important as the throughput estimation technique used in improving the perceived throughput [40], [78].
  - ⋆ Scheduling policy can be used to reduce the power consumption a mobile device [51], [138].
  - ⋆ A good ABR scheme requires a main factor and at least one adjustment factor.
- *Scheduling function:*
  - ⋆ Employing the progressive dispatch at the ramping-up stage makes it easier for the client's TCP throughput to converge [40], [59].
  - ⋆ Parallel scheduling policy can be used to improve system utilisation [142], [143].
- *Adaptation function:*
  - ⋆ The technique used in realising the adaptation logic has no significant impact on the performance of an ABR scheme [80].
  - ⋆ The simpler the adaptation logic, the better [80].
  - ⋆ To perform effectively, an adaptation logic requires the input of the evolving state of QoE metrics in addition to the conventional resource metrics.
- *External factors:*
  - ⋆ The use of CDN can be a source of delay and reduction in throughput [19], [205].
  - ⋆ Contextual information are excellent source of TCP throughput improvement [91], [197]–[199].
  - ⋆ Chunk size and nature has a significant impact on many QoE metrics [84], [179], [189].

B. Current Challenges

Feedback about the evolving state of the quality of video transmission is certainly a prerequisite to building an effective ABR algorithms. One of the most urgent challenges currently, 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. Another challenge that does require the attention of ABR researcher is modelling the relationship between the various components of an ABR. For example, when building a buffer-based player, an ABR algorithm designer requires the exact relationship between video quality and the buffer state changes.

The size and the nature of video chunks can affect a number of system and QoE level parameters. More work is needed in formalising the relationship between the various parameters and chunk size. This can help content providers to optimise the video chunks presented to ABR clients, and in the process helps in reducing storage requirements as well as traffic in the network. Using the contextual information to improve the performance of ABR is still in its infancy. However, this promising field requires the understanding of the context information that is most relevant to the effectiveness of the ABR scheme. The modelling and analysis of the interrelation between the identified elements, and the various subsystems of the ABR module, is also desirable. With these an ABR designer can build a service that is both truly adaptive and effective.

XI. Conclusion

In the past five years, tremendous amount of effort has been put in standardising and improving the HTTP adaptive streaming (HAS). However, little or no effort has been dedicated to systematically analysing and structuring the research space and different solutions. In this survey, we present a comprehensive review of the client-side adaptive bitrate selection (ABR), the part of the HAS technology that decides the profile and schedule of a chunk to be downloaded by a video streaming client. The paper starts by presenting a framework that divides an ABR module into three subcomponents, i.e., resource estimation module, chunk scheduling function, and adaptation logic. Then an exposition of the benefits, issues, and challenges of locating, monitoring and measuring various resources (e.g., throughput) that different ABR schemes rely on for their decision-making is presented. Furthermore, the paper classifies the chunk-scheduling algorithms into sequential and parallel schedules and then discusses where and when it is most appropriate to employ either of them. Because the technique used in implementing rate adaptation dictates the limit of achievable optimisation, the paper presented a detailed discussion of the various approaches e.g., machine learning or control theory, used in designing and implementing ABR in literature. How these subcomponents interact with each other is then covered. We then concluded by a talk on other relevant aspects related to content nature, CDN, and operating context, which are known to have direct impact on the efficiency of an ABR scheme.

References

[1] 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," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 269–281, Mar. 2001.

[2] "Sandvine global Internet phenomena report," White Paper, Sandvine, Waterloo, ON, Canada, Jun. 2013. [Online]. Available: http://www.sandvine.com/downloads/documents/Phenomena_1H_2013/Sandvine_Global_Internet_Phenomena_Report_1H_2013.pdf

[3] "Cisco visual networking index: Forecast and methodology, 2014–2019," White Paper, Cisco, San Jose, CA, USA, Apr. 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

[4] A. Begen, T. Akgul, and M. Baugher, "Watching video over the Web: Part 1: Streaming protocols," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 54–63, Mar./Apr. 2011.

[5] S. Blake *et al.*, "An architecture for differentiated services," Internet Eng. Task Force, Fremont, CA, USA, RFC 2475, Dec. 1998.

[6] N. Bouten *et al.*, "Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, May 2014, pp. 1–7.

[7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[8] H. Ketmaneechairat, "A survey and comparison of some popular IPTV applications," in *Proc. 8th Int. Conf. Comput. Technol. Inf. Manag. (ICCM)*, vol. 1. Seoul, South Korea, 2012, pp. 58–63.

[9] T. Hossfeld and K. Leibnitz, "A qualitative measurement survey of popular Internet-based IPTV systems," in *Proc. 2nd Int. Conf. Commun. Electron. (ICCE)*, 2008, pp. 156–161.

[10] M. Ellis, "Understanding the performance of Internet video over residential networks," Ph.D. dissertation, School Comput. Sci., Univ. at Glasgow, Glasgow, U.K., 2012.

[11] M. Van der Schaar and P. A. Chou, *Multimedia Over IP and Wireless Networks: Compression, Networking, and Systems*. Boston, MA, USA: Academic Press, 2011.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "A transport protocol for real-time applications," Internet Eng. Task Force, Fremont, CA, USA, RFC 3550, Jul. 2003.

[13] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," Internet Eng. Task Force, Fremont, CA, USA, RFC 2326, 1998.

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

[15] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[16] 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*. Cham, Switzerland: Springer, 2014, pp. 136–150.

[17] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via TCP: An analytic performance study," in *Proc. 12th Annu. ACM Int. Conf. Multimedia*, New York, NY, USA, 2004, pp. 908–915.

[18] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia, "Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies," *Comput. Netw.*, vol. 81, pp. 320–332, Apr. 2015.

[19] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Santa Clara, CA, USA, 2011, pp. 169–174.

[20] L. Popa, A. Ghodsi, and I. Stoica, "HTTP as the narrow waist of the future Internet," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, Monterey, CA, USA, 2010, p. 6.

[21] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, 2004, pp. 41–54.

[22] *H.265/HEVC Ratification and 4K Video Streaming*. (Jan. 2014). [Online]. Available: https://blogs.iis.net/alexzam/archive/2013/01/28/h-265-hevc-ratification-and-4k-video-streaming.aspx

[23] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[24] B. Bross, H. Schwarz, and D. Marpe, "The new high-efficiency video coding standard," *SMPTE Motion Imag. J.*, vol. 122, no. 4, pp. 25–35, 2013.

[25] Y. Shuai, G. Petrovic, and T. Herfet, "Server-driven rate control for adaptive video streaming using virtual client buffers," in *Proc. IEEE 4th Int. Conf. Consum. Electron. Berlin (ICCE-Berlin)*, Berlin, Germany, 2014, pp. 45–49.

[26] S. Wilk, D. Stohr, and W. Effelsberg, "VAS: A video adaptation service to support mobile video," in *Proc. 25th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Portland, OR, USA, 2015, pp. 37–42.

[27] S. Wilk and W. Effelsberg, "The content-aware video adaptation service for mobile devices," in *Proc. 7th Int. Conf. Multimedia Syst. (MMSys)*, Klagenfurt, Austria, 2016, pp. 1–4.

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

[29] Y. Sani, A. Mauthe, and C. Edwards, "Modelling video rate evolution in adaptive bitrate selection," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Miami, FL, USA, Dec. 2015, pp. 89–94.

[30] G. Cofano, L. De Cicco, and S. Mascolo, "A control architecture for massive adaptive video streaming delivery," in *Proc. Workshop Design Qual. Deployment Adapt. Video Streaming*, Sydney, NSW, Australia, 2014, pp. 7–12.

[31] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla, "SDN based QoE optimization for HTTP-based adaptive video streaming," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Miami, FL, USA, Dec. 2015, pp. 120–123.

[32] "MPEG DASH specification (ISO/IEC 23009-1:2012) dynamic adaptive streaming over HTTP (DASH)—Part 1: Media presentation description and segment formats," ISO/IEC Standard 23009-1:2012, Mar. 2012.

[33] A. Zambelli, "IIS smooth streaming technical overview," Microsoft Corporat., Redmond, WA, USA, Tech. Rep., 2009.

[34] D. Robinson, "Live streaming ecosystems," in *Advanced Content Delivery, Streaming, and Cloud Services*. Hoboken, NJ, USA: Wiley, 2014, pp. 33–49.

[35] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011.

[36] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Santa Clara, CA, USA, 2011, pp. 157–168.

[37] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, Nice, France, 2012, pp. 109–120.

[38] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming," in *Proc. ACM SIGCOMM Workshop Future Human Centric Multimedia Netw.*, Hong Kong, 2013, pp. 9–14.

[39] J. Research. (Jun. 2013). *The Importance of Delivering a Great Online Video Experience*. [Online]. Available: http://www.akamai.com/dl/reports/jupiter_onlinevideoexp.pdf

[40] T.-Y. Huang, "A buffer-based approach to video rate adaptation," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2014.

[41] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proc. 4th Workshop Mobile Video*, 2012, pp. 37–42.

[42] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proc. 22nd Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Toronto, ON, Canada, 2012, pp. 9–14.

[43] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 20–27, Apr. 2012.

[44] 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 J. Wireless Commun. Netw.*, vol. 2014, no. 1, p. 216, 2014.

[45] Z. Li *et al.*, "Streaming video over HTTP with consistent quality," in *Proc. 5th ACM Multimedia Syst. Conf.*, Singapore, 2014, pp. 248–258.

[46] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH system," in *Proc. 3rd Multimedia Syst. Conf.*, Chapel Hill, NC, USA, 2012, pp. 11–22.

[47] R. Pantos, "HTTP live streaming draft-pantos-HTTP-live-streaming-11," Internet Draft, Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-11

[48] X. Yin, V. Sekar, and B. Sinopoli, "Toward a principled framework to design dynamic adaptive streaming algorithms over HTTP," in *Proc. 13th ACM Workshop Hot Topics Netw.*, Los Angeles, CA, USA, 2014, p. 9.

[49] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, 2012, pp. 97–108.

[50] X. Qiu *et al.*, "Optimizing HTTP-based adaptive video streaming for wireless access networks," in *Proc. 3rd IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, Beijing, China, 2010, pp. 838–845.

[51] X. Li, M. Dong, Z. Ma, and F. C. A. Fernandes, "GreenTube: Power optimization for mobile videostreaming via dynamic cache management," in *Proc. 20th ACM Int. Conf. Multimedia (MM)*, Nara, Japan, 2012, pp. 279–288.

[52] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two decades of Internet video streaming: A retrospective view," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1s, pp. 1–20, Oct. 2013.

[53] N. Zong, "Survey and gap analysis for HTTP streaming standards and implementations," Internet Eng. Task Force, Fremont, CA, USA, Netw. Working Group, 2011.

[54] M. Seufert *et al.*, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[55] H. Luo and M.-L. Shyu, "Quality of service provision in mobile multimedia—A survey," *Human Centric Comput. Inf. Sci.*, vol. 1, no. 1, pp. 1–15, 2011.

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

[57] M. A. Hoque, M. Siekkinen, J. K. Nurminen, M. Aalto, and S. Tarkoma, "Mobile multimedia streaming techniques: QoE and energy saving perspective," *Pervasive Mobile Comput.*, vol. 16, pp. 96–114, Jan. 2015.

[58] *MoveNetworks*. (Jun. 2013). [Online]. Available: http://www.movenetworks.com/history.html

[59] Z. Li *et al.*, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.

[60] "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," 3rd Gener. Partnership Project, Sophia Antipolis, France, Tech. Rep. V10.3.0, Mar. 2013.

[61] "3OIPF release 2 release 2 specification HTTP adaptive streaming [V2....0]," Open IPTV Forum, Sophia Antipolis, France, Tech. Rep. V2.3, Sep. 2010.

[62] *OSMF Player*, Adobe, Mountain View, CA, USA, Jun. 2013. [Online]. Available: http://www.osmf.org

[63] F. Wamser *et al.*, "YoMoApp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Paris, France, Jun. 2015, pp. 239–243.

[64] L. Toni *et al.*, "Optimal selection of adaptive streaming representations," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 2s, pp. 1–26, Feb. 2015, doi: 10.1145/2700294.

[65] T. Stockhammer, "Dynamic adaptive streaming over HTTP—Standards and design principles," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Santa Clara, CA, USA, 2011, pp. 133–144.

[66] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, Lucca, Italy, Jun. 2011, pp. 1–8.

[67] V. Swaminathan and S. Wei, "Low latency live video streaming using HTTP chunked encoding," in *Proc. IEEE 13th Int. Workshop Multimedia Signal Process. (MMSP)*, Hangzhou, China, Oct. 2011, pp. 1–6.

[68] S. Wei and V. Swaminathan, "Low latency live video streaming over HTTP 2.0," in *Proc. Netw. Oper. Syst. Support Digit. Audio Video Workshop (NOSSDAV)*, Singapore, 2014, pp. 37–42.

[69] R. Huysegems *et al.*, "HTTP/2-based methods to improve the live experience of adaptive streaming," in *Proc. 23rd Annu. ACM Conf. Multimedia Conf. (MM)*, Brisbane, QLD, Australia, 2015, pp. 541–550.

[70] M. Xiao, V. Swaminathan, S. Wei, and S. Chen, "Evaluating and improving push based video streaming with HTTP/2," in *Proc. 26th Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Klagenfurt, Austria, 2016, pp. 1–6.

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

[72] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP adaptive streaming: Friend or foe?" in *Proc. Netw. Oper. Syst. Support Digit. Audio Video Workshop*, Singapore, 2014, p. 31.

[73] *SLExtensions Adaptive Streaming*. (Mar. 2014). [Online]. Available: https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

[74] A. Rao *et al.*, "Network characteristics of video streaming traffic," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, Tokyo, Japan, 2011, p. 25.

[75] 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 *Proc. ACM Conf. SIGCOMM (SIGCOMM)*, Chicago, IL, USA, 2014, pp. 187–198.

[76] Y. Sani, A. Mauthe, and C. Edwards, "On the trajectory of video quality transition in HTTP adaptive video streaming," *Multimedia Syst.*, pp. 1–14, Jun. 2017.

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

[78] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. ACM Conf. Internet Meas. Conf.*, Boston, MA, USA, 2012, pp. 225–238.

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

[80] C. Timmerer, M. Maiero, and B. Rainer, "Which adaptation logic? An objective and subjective performance evaluation of HTTP-based adaptive media streaming systems," *arXiv*, 2016.

[81] C. Kreuzberger, B. Rainer, H. Hellwagner, L. Toni, and P. Frossard, "A comparative study of DASH representation sets using real user characteristics," in *Proc. 26th Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Klagenfurt, Austria, 2016, pp. 1–6.

[82] X. Liu *et al.*, "A case for a coordinated Internet video control plane," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Helsinki, Finland, 2012, pp. 359–370.

[83] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using OpenFlow-assisted adaptive video streaming," in *Proc. ACM SIGCOMM Workshop Future Human Centric Multimedia Netw. (FhMN)*, Hong Kong, 2013, pp. 15–20.

[84] C. Liu, I. Bouazizi, and M. Gabbouj, "Segment duration for rate adaptation of adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Barcelona, Spain, 2011, pp. 1–4.

[85] S. Gouache, G. Bichot, A. Bsila, and C. Howson, "Distributed & adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Barcelona, Spain, 2011, pp. 1–6.

[86] G. Tian and Y. Liu, "On adaptive HTTP streaming to mobile devices," in *Proc. 20th Int. Packet Video Workshop (PV)*, San Jose, CA, USA, Dec. 2013, pp. 1–8.

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

[88] T. C. Thang, A. T. Pham, H. X. Nguyen, P. L. Cuong, and J. W. Kang, "Video streaming over HTTP with dynamic resource prediction," in *Proc. 4th Int. Conf. Commun. Electron. (ICCE)*, 2012, pp. 130–135.

[89] V. Ramamurthi and O. Oyman, "Link aware HTTP adaptive streaming for enhanced quality of experience," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Atlanta, GA, USA, Dec. 2013, pp. 1675–1680.

[90] V. Ramamurthi, O. Oyman, and J. Foerster, "Using link awareness for HTTP adaptive streaming over changing wireless conditions," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Garden Grove, CA, USA, Feb. 2015, pp. 727–731.

[91] J. Hao, R. Zimmermann, and H. Ma, "GTube: Geo-predictive video streaming over HTTP in mobile environments," in *Proc. 5th ACM Multimedia Syst. Conf. (MMSys)*, Singapore, 2014, pp. 259–270.

[92] R. Kuschnig, I. Kofler, and H. Hellwagner, "Improving Internet video streaming performance by parallel TCP-based request-response streams," in *Proc. 7th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2010, pp. 1–5.

[93] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. 23rd ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Oslo, Norway, 2013, pp. 19–24.

[94] R. Houdaille and S. Gouache, "Shaping HTTP adaptive streams for a better user experience," in *Proc. 3rd Multimedia Syst. Conf. (MMSys)*, Chapel Hill, NC, USA, 2012, pp. 1–9.

[95] B. J. Villa and P. E. Heegaard, "Group based traffic shaping for adaptive HTTP video streaming by segment duration control," in *Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Barcelona, Spain, Mar. 2013, pp. 830–837.

[96] B. J. Villa and P. E. Heegaard, "Improving perceived fairness and QoE for adaptive video streams," in *Proc. 8th Int. Conf. Netw. Services (ICNS)*, 2012, pp. 149–158.

[97] K. Evensen, T. Kupka, D. Kaspar, P. Halvorsen, and C. Griwodz, "Quality-adaptive scheduling for live streaming over multiple access networks," in *Proc. 20th Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Amsterdam, The Netherlands, 2010, pp. 21–26.

[98] K. Evensen *et al.*, "Using bandwidth aggregation to improve the performance of quality-adaptive streaming," *Signal Process. Image Commun.*, vol. 27, no. 4, pp. 312–328, 2012.

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

[100] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, "Buffer-based smooth rate adaptation for dynamic HTTP streaming," in *Proc. Asia–Pac. Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA)*, 2013, pp. 1–9.

[101] H. T. Le, D. V. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, "Buffer-based bitrate adaptation for adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Adv. Technol. Commun. (ATC)*, 2013, pp. 33–38.

[102] R. Trestian, A.-N. Moldovan, O. Ormond, and G.-M. Muntean, "Energy consumption analysis of video streaming to android mobile devices," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2012, pp. 444–452.

[103] M. Hosseini, J. Peters, and S. Shirmohammadi, "Energy-budget-compliant adaptive 3D texture streaming in mobile games," in *Proc. 4th ACM Multimedia Syst. Conf. (MMSys)*, Oslo, Norway, 2013, pp. 1–11.

[104] M. A. Hoque, M. Siekkinen, J. K. Nurminen, and M. Aalto, "Dissecting mobile video services: An energy consumption perspective," in *Proc. IEEE 14th Int. Symp. Workshops World Wireless Mobile Multimedia Netw. (WoWMoM)*, Madrid, Spain, 2013, pp. 1–11.

[105] M. N. Ismail, R. Ibrahim, and M. F. M. Fudzee, "A survey on content adaptation systems towards energy consumption awareness," *Adv. Multimedia*, vol. 2013, p. 3, Jan. 2013.

[106] N. Zsak and C. Wolff, "Impact of video quality and wireless network interface on power consumption of mobile devices," *arXiv preprint arXiv:1407.7667*, 2014.

[107] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is energy consumed in smartphone display applications?" in *Proc. 14th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2013, pp. 1–6.

[108] S. Khan, D. Schroeder, A. El Essaili, and E. Steinbach, "Energy-efficient and QoE-driven adaptive HTTP streaming over LTE," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Istanbul, Turkey, Apr. 2014, pp. 2354–2359.

[109] W. Lee, J. Koo, S. Jin, and S. Choi, "EQ-video: Energy and quota-aware video playback time maximization for smartphones," *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 1045–1048, Jun. 2015.

[110] S. Kim, H. Oh, and C. Kim, "ePF-DASH: Energy-efficient prefetching based dynamic adaptive streaming over HTTP," in *Proc. Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2015, pp. 124–129.

[111] M. Hosseini, A. Wang, and R. Etesami, "Towards energy-aware DASH for mobile video," in *Proc. 7th ACM Int. Workshop Mobile Video (MoVid)*, Portland, OR, USA, 2015, pp. 7–8.

[112] L. Zou, R. Trestian, and G.-M. Muntean, "eDOAS: Energy-aware device-oriented adaptive multimedia scheme for Wi-Fi offload," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Istanbul, Turkey, Apr. 2014, pp. 2916–2921.

[113] J. Chen, A. Ghosh, J. Magutt, and M. Chiang, "QAVA: Quota aware video adaptation," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, Nice, France, 2012, pp. 121–132.

[114] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, 2004, pp. 272–277.

[115] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Netw.*, vol. 17, no. 6, pp. 27–35, Nov./Dec. 2003.

[116] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 97–108, 2007.

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

[118] T. Nguyen and S.-C. S. Cheung, "Multimedia streaming using multiple TCP connections," in *Proc. 24th IEEE Int. Perform. Comput. Commun. Conf. (IPCCC)*, Phoenix, AZ, USA, 2005, pp. 215–223.

[119] M. H. Alizai, O. Landsiedel, J. Á. B. Link, S. Götz, and K. Wehrle, "Bursty traffic over bursty links," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Berkeley, CA, USA, 2009, pp. 71–84.

[120] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, pp. 47–52, 2008.

[121] B. Wang, W. Wei, Z. Guo, and D. Towsley, "Multipath live streaming via TCP: Scheme, performance and benefits," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 5, no. 3, p. 25, 2009.

[122] J. G. Apostolopoulos, W.-T. Tan, and S. J. Wee, "Video streaming: Concepts, algorithms, and systems," HP Lab., Palo Alto, CA, USA, Tech. Rep. HPL-2002-260, 2002.

[123] A. Dua and N. Bambos, "Buffer management for wireless media streaming," in *Proc. Glob. Telecommun. Conf. (GLOBECOM)*, Washington, DC, USA, 2007, pp. 5226–5230.

[124] T. Kim and M. H. Ammar, "Receiver buffer requirement for video streaming over TCP," in *Proc. Electron. Imag.*, San Jose, CA, USA, 2006, Art. no. 607718.

[125] *Akamai Reveals 2 Seconds As the New Threshold of Acceptability for Ecommerce Web Page Response Times*, Akamai Technol. Inc., Cambridge, MA, USA, Sep. 2009. [Online]. Available: http://www.akamai.com/html/about/press/releases/2009/press_091409.html

[126] S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 2001–2014, Dec. 2013.

[127] 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," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 988–1001, Jun. 2015.

[128] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *Int. J. Human Comput. Stud.*, vol. 64, no. 8, pp. 637–647, 2006.

[129] Y. Sani, A. Mauthe, C. Edwards, and M. Mu, "A bio-inspired HTTP-based adaptive streaming player," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Seattle, WA, USA, Jul. 2016, pp. 1–4.

[130] F. Jason and S. Mahadev, "Energy-aware adaptation for mobile applications," *SIGOPS Oper. Syst. Rev.*, vol. 33, no. 5, pp. 48–63, Dec. 1999.

[131] J. Flinn and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Trans. Comput. Syst.*, vol. 22, no. 2, pp. 137–179, May 2004.

[132] S. Chandra and A. Vahdat, "Application-specific network management for energy-aware streaming of popular multimedia formats," in *Proc. USENIX Annu. Tech. Conf. Gen. Track*, 2002, pp. 329–342.

[133] A. Seema, L. Schwoebel, T. Shah, J. Morgan, and M. Reisslein, "WVSNP-DASH: Name-based segmented video streaming," *IEEE Trans. Broadcast.*, vol. 61, no. 3, pp. 346–355, Sep. 2015.

[134] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Using crowd-sourced viewing statistics to save energy in wireless video streaming," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 377–388.

[135] H. Abou-Zeid, H. S. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2013–2026, Jun. 2014.

[136] Y. Chen, B. Zhang, Y. Liu, and W. Zhu, "Measurement and modeling of video watching time in a large-scale Internet video-on-demand system," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2087–2098, Dec. 2013.

[137] D. Johansen *et al.*, "DAVVI: A prototype for the next generation multimedia entertainment platform," in *Proc. 17th ACM Int. Conf. Multimedia*, Beijing, China, 2009, pp. 989–990.

[138] N. Gautam, H. Petander, and J. Noel, "A comparison of the cost and energy efficiency of prefetching and streaming of mobile video," in *Proc. 5th Workshop Mobile Video (MoVid)*, Oslo, Norway, 2013, pp. 7–12.

[139] B. J. Villa, P. E. Heegaard, and A. Instefjord, "Improving fairness for adaptive HTTP video streaming," in *Information and Communication Technologies*. Heidelberg, Germany: Springer, 2012, pp. 183–193.

[140] T. Kupka, P. Halvorsen, and C. Griwodz, "An evaluation of live adaptive HTTP segment streaming request strategies," in *Proc. IEEE 36th Conf. Local Comput. Netw.*, Bonn, Germany, Oct. 2011, pp. 604–612.

[141] D. Kaspar *et al.*, "Enhancing video-on-demand playout over multiple heterogeneous access networks," in *Proc. 7th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2010, pp. 1–5.

[142] D. Kaspar, K. Evensen, P. Engelstad, and A. F. Hansen, "Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2010, pp. 1–5.

[143] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel adaptive HTTP media streaming," in *Proc. IEEE 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2011, pp. 1–6.

[144] M.-N. Garcia *et al.*, "Quality of experience and HTTP adaptive streaming: A review of subjective studies," in *Proc. 6th Int. Workshop Qual. Multimedia Exp. (QoMEX)*, Singapore, 2014, pp. 141–146.

[145] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori, "DASH fast start using HTTP/2," in *Proc. 25th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Portland, Oregon, 2015, pp. 25–30.

[146] S. Wei and V. Swaminathan, "Cost effective video streaming using server push over HTTP 2.0," in *Proc. IEEE 16th Int. Workshop Multimedia Signal Process. (MMSP)*, Jakarta, Indonesia, 2014, pp. 1–5.

[147] M. Mu, S. Simpson, A. Farshad, Q. Ni, and N. Race, "User-level fairness delivered: Network resource allocation for adaptive video streaming," in *Proc. IEEE 23rd Int. Symp. Qual. Service (IWQoS)*, Portland, OR, USA, Jun. 2015, pp. 85–94.

[148] L. De Cicco and S. Mascolo, "An experimental investigation of the Akamai adaptive video streaming," in *HCI in Work and Learning, Life and Leisure*. Heidelberg, Germany: Springer, 2010, pp. 447–464.

[149] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Santa Clara, CA, USA, 2011, pp. 145–156.

[150] C. Zhou, X. Zhang, and Z. Guo, "A control theory based rate adaption scheme for DASH over multiple servers," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Kuching, Malaysia, 2013, pp. 1–6.

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

[152] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, London, U.K., 2015, pp. 325–338.

[153] N. Bouten *et al.*, "QoE optimization through in-network quality adaptation for HTTP adaptive streaming," in *Proc. 8th Int. Conf. Workshop Syst. Virtual. Manag. (SVM) Netw. Service Manag. (CNSM)*, Las Vegas, NV, USA, Oct. 2012, pp. 336–342.

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

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

[156] P. Xiong *et al.*, "NBS: A network-bandwidth-aware streaming version switcher for mobile streaming applications under fuzzy logic control," in *Proc. IEEE 1st Int. Conf. Mobile Services (MS)*, Honolulu, HI, USA, 2012, pp. 48–55.

[157] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados, "A control-based algorithm for rate adaption in MPEG-DASH," in *Proc. 5th Int. Conf. Inf. Intell. Syst. Appl. (IISA)*, Chania, Greece, Jul. 2014, pp. 438–442.

[158] A. Sobhani, A. Yassine, and S. Shirmohammadi, "A fuzzy-based rate adaptation controller for DASH," in *Proc. 25th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Portland, Oregon, 2015, pp. 31–36.

[159] M. Claeys *et al.*, "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Proc. Conf. Auton. Agents Multiagent Syst.*, May 2013, pp. 30–37.

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

[161] 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 *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2015, pp. 1–6.

[162] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, Jan. 2013, pp. 127–128.

[163] 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 *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2015, pp. 131–138.

[164] 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*. Boston, MA, USA: Springer, 2008, pp. 185–215.

[165] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 693–705, Apr. 2014.

[166] S. Basso, A. Servetti, E. Masala, and J. C. De Martin, "Measuring DASH streaming performance from the end users perspective using neubot," in *Proc. 5th ACM Multimedia Syst. Conf. (MMSys)*, Singapore, 2014, pp. 1–6.

[167] S. Di, Y. Zhao, C. Li, and Y. Guo, "An energy-aware chunk selection mechanism in HTTP video streaming," in *Proc. IEEE 8th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2016, pp. 1–5.

[168] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan, "Empirical evaluation of HTTP adaptive streaming under vehicular mobility," in *Proc. Int. Conf. Res. Netw.*, 2011, pp. 92–105.

[169] J. Lievens, S. M. Satti, N. Deligiannis, P. Schelkens, and A. Munteanu, "Optimized segmentation of H.264/AVC video for HTTP adaptive streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, May 2013, pp. 1312–1317.

[170] *HTTP Dynamic Streaming on the Adobe Flash Platform: Enabling High-Quality, Network-Efficient HTTP Streaming for Media Delivery*. (Aug. 2010). [Online]. Available: https://bugbase.adobe.com/index.cfm?event=file.view&id=2943064 seqNum=6 name=httpdynamicstreaming_wp_ue.pdf

[171] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proc. 3rd ACM Multimedia Syst. Conf. (MMSys)*, 2012, pp. 89–94.

[172] *YouTube MPEG-DASH/Media Source Demo*. (Aug. 2015). [Online]. Available: http://dash-mse-test.appspot.com/media.html

[173] *Ultra High Definition HEVC DASH Data Set*. (Aug. 2015). [Online]. Available: http://download.tsi.telecom-paristech.fr/gpac/dataset/dash/uhd/

[174] J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Datasets for AVC (H.264) and HEVC (H.265) evaluation of dynamic adaptive streaming over HTTP (DASH)," in *Proc. ACM 7th Int. Conf. Multimedia Syst. (MMSys)*, Klagenfurt, Austria, 2016, pp. 1–6.

[175] *17 Free & Public MPEG-DASH and HLS Example Test Streams and Datasets*. (Apr. 2017). [Online]. Available: https://bitmovin.com/mpeg-dash-hls-examples-sample-streams/

[176] U. Jeong and K. Chung, "Video quality adaptation to improve the quality of experience in DASH environments," *Int. J. Comput. Sci. Netw. Security*, vol. 14, no. 8, pp. 22–29, 2014.

[177] R. Huysegems, B. De Vleeschauwer, T. Wu, and W. Van Leekwijck, "SVC-based HTTP adaptive streaming," *Bell Labs Tech. J.*, vol. 16, no. 4, pp. 25–41, 2012.

[178] Y. S. de la Fuente *et al.*, "iDASH: Improved dynamic adaptive streaming over HTTP using scalable video coding," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, San Jose, CA, USA, 2011, pp. 257–264.

[179] Y. Sánchez, C. Hellge, T. Schierl, W. Van Leekwijck, and Y. Le Louédec, "Scalable video coding based DASH for efficient usage of network resources," in *Proc. 3rd W3C Web TV Workshop*, 2011, pp. 19–20.

[180] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[181] H. Schwarz and M. Wien, "The scalable video coding extension of the H.264/AVC standard," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 135–141, Mar. 2008.

[182] I. Unanue *et al.*, "A tutorial on H.264/SVC scalable video coding and its tradeoff between quality, coding efficiency and performance," *Recent Adv. Video Coding*, vol. 13, pp. 1–24, Jul. 2011.

[183] K. Tappayuthpijarn, T. Stockhammer, and E. Steinbach, "HTTP-based scalable video streaming over mobile networks," in *Proc. 18th IEEE Int. Conf. Image Process. (ICIP)*, Brussels, Belgium, 2011, pp. 2193–2196.

[184] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proc. ACM 3rd Multimedia Syst. Conf.*, Chapel Hill, NC, USA, 2012, pp. 167–172.

[185] M. Grafl, C. Timmerer, H. Hellwagner, W. Cherif, and A. Ksentini, "Evaluation of hybrid scalable video coding for HTTP-based adaptive media streaming with high-definition content," in *Proc. IEEE 14th Int. Symp. Workshops World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2013, pp. 1–7.

[186] J. Famaey *et al.*, "On the merits of SVC-based HTTP adaptive streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, 2013, pp. 419–426.

[187] C. Müller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer, "Using scalable video coding for dynamic adaptive streaming over HTTP in mobile environments," in *Proc. 20th Eur. Signal Process. Conf. (EUSIPCO)*, Bucharest, Romania, 2012, pp. 2208–2212.

[188] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, "Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, 2013, pp. 1318–1323.

[189] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over HTTP with scalable video coding," in *Proc. ACM 3rd Multimedia Syst. Conf.*, Chapel Hill, NC, USA, 2012, pp. 149–154.

[190] H. Kalva, V. Adzic, and B. Furht, "Comparing MPEG AVC and SVC for adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, 2012, pp. 158–159.

[191] K. Mitra, A. Zaslavsky, and C. Åhlund, "Context-aware QoE modelling, measurement, and prediction in mobile computing systems," *IEEE Trans. Mobile Comput.*, vol. 14, no. 5, pp. 920–936, May 2015.

[192] F. Metzger, E. Liotou, C. Moldovan, and T. Hoßfeld, "TCP video streaming and mobile networks: Not a love story, but better with context," *Comput. Netw.*, vol. 109, pp. 246–256, Nov. 2016.

[193] T. Hoßfeld *et al.*, "Can context monitoring improve QoE? A case study of video flash crowds in the Internet of services," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2015, pp. 1274–1277.

[194] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Proc. 6th Int. Workshop Qual. Multimedia Exp. (QoMEX)*, Singapore, 2014, pp. 111–116.

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

[196] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in high-speed mobility using bandwidth maps," *IEEE Trans. Mobile Comput.*, vol. 11, no. 4, pp. 603–617, Apr. 2012.

[197] X. K. Zou *et al.*, "Can accurate predictions improve video streaming in cellular networks?" in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl. (HotMobile)*, Santa Fe, NM, USA, 2015, pp. 57–62.

[198] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Trans. Multimedia Comput. Commun. Appl. (TOMM)*, vol. 8, no. 3, p. 24, 2012.

[199] D. Han *et al.*, "MASERATI: Mobile adaptive streaming based on environmental and contextual information," in *Proc. 8th ACM Int. Workshop Wireless Netw. Testbeds Exp. Eval. Characterization (WiNTECH)*, Miami, FL, USA, 2013, pp. 33–40.

[200] R. Dubin *et al.*, "Adaptation logic for HTTP dynamic adaptive streaming using geo-predictive crowdsourcing for mobile users," *Multimedia Syst.*, pp. 1–13, Feb. 2016.

[201] B. Taani and R. Zimmermann, "Spatio-temporal analysis of bandwidth maps for geo-predictive video streaming in mobile environments," in *Proc. ACM Multimedia Conf.*, Amsterdam, The Netherlands, 2016, pp. 888–897.

[202] E. Liotou *et al.*, "Enriching HTTP adaptive streaming with context awareness: A tunnel case study," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[203] M. Pathan, "Cloud-based content delivery and streaming," in *Advanced Content Delivery, Streaming, and Cloud Services*. Hoboken, NJ, USA: Wiley, 2014, pp. 1–31.

[204] J. Dilley *et al.*, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep./Oct. 2002.

[205] F. Zhou, S. Ahmad, E. Buyukkaya, R. Hamzaoui, and G. Simon, "Minimizing server throughput for low-delay live streaming in content delivery networks," in *Proc. ACM 22nd Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Toronto, ON, Canada, 2012, pp. 65–70.

[206] J. Famaey, S. Latré, R. van Brandenburg, M. O. van Deventer, and F. De Turck, "On the impact of redirection on HTTP adaptive streaming services in federated CDNs," in *Proc. IFIP Int. Conf. Auton. Infrastruct. Manag. Security*, 2013, pp. 13–24.

[207] R. V. Brandenburg, F. L. Faucheur, O. V. Deventer, and K. Leung, "Models for HTTP-adaptive-streaming-aware content distribution network interconnection (CDNI)," Internet Eng. Task Force, Fremont, CA, USA, RFC 6983, 2013.

[208] K. Liang, J. Hao, R. Zimmermann, and D. K. Y. Yau, "Integrated prefetching and caching for adaptive video streaming over HTTP: An online approach," in *Proc. 6th ACM Multimedia Syst. Conf.*, 2015, pp. 142–152.

[209] P. Juluri and D. Medhi, "Cache'n DASH: Efficient caching for DASH," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 599–600, 2015.
[210] L. Peterson, B. Davie, and R. van Brandenburg, "Framework for content distribution network interconnection (CDNI)," Internet Eng. Task Force, Fremont, CA, USA, RFC 7336, 2014.
[211] E. Thomas *et al.*, "Applications and deployments of server and network assisted DASH (SAND)," in *Proc. IBC*, 2016, p. 22.
[212] "Netflix open connect," White Paper, Netflix, Los Gatos, CA, USA, Jan. 2017. [Online]. Available: https://openconnect.netflix.com/en/
[213] *Netflix: We're the Ones Throttling Video Speeds on AT&T and Verizon*. (Jan. 2017). [Online]. Available: https://www.cnet.com/news/netflix-admits-throttling-video-speeds-on-at-t-verizon/

Yusuf Sani received the bachelor's degree in computing and information systems from the University of London, U.K., and the master's degree in distributed computing from Universiti Putra Malaysia. He is currently pursuing the Ph.D. degree at Lancaster University. His research interests lie at the intersection of mobile networking and multimedia systems. He is also a Lecturer at the Kano University of Science and Technology, Wudil, Nigeria.

Andreas Mauthe is a Reader in networked systems with Lancaster University. He has been leading networking and systems projects in academia and industry for over 15 years. He has authored over 150 peer-reviewed papers. His research focus is in network management, and autonomic and resilient systems, alongside multimedia systems, content networking, and adaptive networks. He is an Associate Editor of *Multimedia Systems*.

Christopher Edwards received the B.Sc. (First Class Hons.) degree in computer studies from Liverpool John Moores University, U.K., in 1995 and the Ph.D. degree in computer science from Lancaster University, U.K., in 2000. Since 2000, he has been involved in leading Lancaster's input into various EPSRC and EU funded initiatives. He is a Senior Lecturer with the School of Computing and Communications, Lancaster University, where he teaches in the areas of advanced network and operating systems. He is also the Postgraduate Studies Associate Dean with the Faculty of Science and Technology, and the Education Theme Lead of the Lancaster University Data Science Institute. His research interests are in the areas of network mobility, and the efficient provision of rural Internet access.