A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP

Jonathan Kua, *Member, IEEE*, Grenville Armitage, *Member, IEEE* and Philip Branch, *Member, IEEE*

***Abstract*—With companies such as Netflix and YouTube accounting for more than 50% of the peak download traffic on North American fixed networks in 2015, video streaming represents a significant source of Internet traffic. Multimedia delivery over the Internet has evolved rapidly over the past few years. The last decade has seen video streaming transitioning from User Datagram Protocol (UDP) to Transmission Control Protocol (TCP)-based technologies. Dynamic Adaptive Streaming over HTTP (DASH) has recently emerged as a standard for Internet video streaming. A range of rate adaptation mechanisms are proposed for DASH systems in order to deliver video quality that matches the throughput of dynamic network conditions for a richer user experience. This survey paper looks at emerging research into the application of client-side, server-side and in-network rate adaptation techniques to support DASH-based content delivery. We provide context and motivation for the application of these techniques and review significant works in the literature from the past decade. These works are categorised according to the feedback signals used and the end-node that performs or assists with the adaptation. We also provide a review of several notable video traffic measurement and characterisation studies and outline open research questions in the field.**

***Index Terms*—DASH, HTTP, TCP, ABR, QoE, adaptive multimedia streaming, traffic measurement**

I. Introduction

In recent years, we have seen the rapid convergence of various multimedia services such as traditional TV, Internet Protocol TV (IPTV), video conferencing, video-on-demand, live and mobile streaming services. With companies such as Netflix and YouTube accounting for more than 50% of the peak download traffic on fixed networks in North America in 2015 [1], video streaming represents a significant source of Internet traffic. In the past decade, the Internet has become a standard medium for multimedia delivery. The Hypertext Transfer Protocol (HTTP) [2] on top of Transmission Control Protocol (TCP) [3] has become the primary protocol for multimedia content delivery over the Internet, also widely known as over-the-top (OTT) or Internet Protocol (IP)-based content delivery. Cisco Visual Networking Index forecasts IP video traffic to be 82% of all consumer traffic and Content Delivery Networks (CDN) will carry two-thirds of all Internet traffic by 2020 [4].

Real-time multimedia delivery has tight latency constraints, and data arriving too late is essentially useless. Efficient media compression creates interdependence between packet contents and codecs, so packet losses and late arrivals of video data can be detrimental. When combined with the inherent nature of network environments and transport protocol behaviours, effective multimedia delivery presents many challenges.

Applications run on top of transport protocols and real-time multimedia applications ideally trust the transport layer to minimise induced delays and deliver data with an appropriate degree of reliability and timeliness. Over the years, message-oriented transport protocols, such as Stream Control Transmission Protocol (SCTP) [5] and Datagram Congestion Control Protocol (DCCP) [6] have been thought suitable, but the existence of Network Address Translations (NAT) [7], [8], firewalls, and other middleboxes has led to a range of known deployment challenges [9], [10], [11]. Hence, most early work focused on enhancing User Datagram Protocol (UDP) [12] for multimedia delivery. On the other hand, although TCP prefers reliability to timeliness and its congestion control tends to induce high queuing delays, it traverses any network path that supports regular HTTP-based communication. Therefore, in recent years TCP has rapidly supplanted UDP as the standard for multimedia delivery.

Standards bodies have been developing various technologies for multimedia transport and encapsulation over the years, such as digital broadcasting, audio and video transport over the Internet and streaming to mobile devices. For example, the Moving Picture Experts Group (MPEG) developed MPEG-2 Transport Stream and International Organisation for Standardisation (ISO) base media file format. The Internet Engineering Task Force (IETF), Institute of Electrical and Electronics Engineers (IEEE), 3rd Generation Partnership Project (3GPP) have also provided many protocols for multimedia content delivery packetised by MPEG technologies. Recently, MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [13] has become a standard that aims to provide an uninterrupted video streaming service to users with dynamic network conditions and heterogeneous devices using an application layer Adaptive Bitrate (ABR)[1] algorithm. The main goal of ABR algorithms is to prevent client's playout buffer under-run, while maximising the perceived Quality of Experience (QoE) of the user by adapting to the dynamically changing network conditions. Some deployment examples in the industry are Microsoft Smooth Streaming [16], Adobe HTTP Dynamic Streaming [17] and Apple's HTTP Live Streaming [18].

The authors are with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia. Email: {jtkua, garmitage, pbranch}@swin.edu.au

[1]The acronym ABR here is not to be confused with ABR (*average bitrate*) encoding used by LAME [14] developers to refer to a type of variable bit rate encoding; or the ABR (*Available Bit Rate*) service category used in early ATM networks [15].

This paper provides a survey of key rate adaptation techniques in the literature. DASH specifications do not enforce any particular adaptation algorithm, which provides flexibility for ABR development. We categorise these rate adaptation techniques in terms of the feedback signals they use and the end-node that performs or assists the adaptation.

The rest of this paper is structured as follows. Section II summarises the history of video delivery over packet networks and how DASH emerged as preferred choice for content delivery. Section III describes DASH architectural design, standardisation and applications. Section IV surveys client-side throughput-based, buffer-based and hybrid/control theory-based techniques in research literature. Section V reviews proposed techniques for server-side, transport layer and network level solutions to assist with DASH rate adaptation. Section VI reviews several notable video streaming traffic measurement and characterisation studies. Section VII offers concluding remarks and outlines open research questions in the field for potential future work.

II. BACKGROUND

Here we cover the evolution of video delivery over packet networks and associated technologies.

A. Video delivery in IP networks

Video frames should be played between 24-30 frames per second to create the illusion of motion. Video compression algorithms carry out both intra and inter-frame compression, which have temporal dependencies, resulting in Intra (I), Bidirectional (B) and Predicted (P) frames. I frames are largest because they only use intra frame compression, whereas B and P frames are smaller because they use previous I frames for size reduction [19]. The variability in encoded bits per second leads to Variable Bit Rate (VBR)[2] video, as used by the ITU (International Telecommunication Union) H.264 and MPEG-4 video coding standards.

The VBR encoded video is then transmitted into the Internet. Since the Internet does not provide a constant, guaranteed bandwidth for the video stream, the network can only support the video bitrate on a best-effort basis. If the network bandwidth is not sufficient to support the video bitrate, then the decoder at the client-end starts to consume the video data at a greater rate than at which new data is being received from the network. The decoder eventually runs out of video data to decode, which results in a screen freeze (video stalls or rebuffering events). In order to avoid this consequence without having to introduce costly and complex guaranteed bandwidth mechanisms, the following solutions have been used to try to match the video bitrate to the available network bandwidth [19]:

- Using a playout buffer: Short-term variations in network throughput can be overcome by using a playout buffer. The video player can decode the pre-fetched data stored in the playout buffer.
- Transcoding-based solutions: These solutions change one or more parameters of the raw video data compression algorithm to vary the resulting bitrate. Examples include varying the video resolution, compression ratio, or frame rate. However, this process is computationally intensive and requires complex hardware support.
- Scalable encoding solutions: These solutions are implemented by processing the encoded video data. Hence, the encoded video can be adapted on-the-fly by using the scalability features of the encoder. Some techniques include adapting the picture resolution or frame rate (by exploiting the spatial or temporal scalability in the encoded data). However, specialised servers are required to implement these solutions.
- Stream switching solutions: This technique is the simplest to implement and used in CDNs. Raw video data is pre-processed to produce multiple encoded streams, each at a different bitrate, resulting in multiple versions of the same content. A client-side adaptive algorithm is then used to select the most appropriate rate given the network conditions during transmission. These solutions do not require specialised servers and use the least processing power. However, more storage and finer granularity of encoded bitrates are required to enable the client to optimise its selection.

Considering feasibility of deployments, the industry has settled on using playout buffers and stream switching solutions. In order to avoid buffer under-run, the video server has to use an appropriate sending rate. In some of the early work on video transport, protocols such as Rate Adaptation Protocol (RAP) [20] and TCP Friendly Rate Control (TFRC) [21] were defined on top of the transport layer that put the sender in charge of varying the sending rate (and consequently the video rate) based on feedback being received from either the network or the receiver, forming a combination of congestion control and flow control. RAP used a TCP-like additive increase/multiplicative decrease (AIMD) scheme. TFRC uses an additive increase/additive decrease (AIAD) scheme to adjust the server's sending rate by estimating the path's throughput based on TCP square root formula – using the path's Round Trip Time (RTT) and packet loss rate.

B. Video streaming evolution

The Internet was not originally designed for the sustained delivery of modern bandwidth-intensive applications such as high quality multimedia streaming. The fundamental difference between traditional data traffic and video traffic is the real-time constraints on video traffic. Most of the early work on packet video transmission focused on providing real-time transmission with techniques that support resource reservations and Quality of Service (QoS), such as Resource ReSerVation Protocol (RSVP) [22] and Integrated Services (IntServ) [23]. Other protocols such as Real-time Transport Protocol (RTP) [24], Real Time Streaming Protocol (RTSP) [25], Session Description Protocol (SDP) [26], RTP Control Protocol (RTCP) [27] were developed over the years to support real-time streaming over UDP and configure/control

[2]Variable bit rate (VBR) encoding methods generate variable amounts of output data per time segment.

the end systems that support video streams. However, these techniques have issues in traversing NATs and firewalls, and they require dedicated servers and network infrastructure, which increases deployment costs and added complexities.

TCP is a reliable protocol which guarantees the delivery of data. However, this reliability comes at the expense of variable delay as senders wait for acknowledgments (ACK) and retransmit lost data. Since video is often delay intolerant and often does not need high reliability to be acceptable, TCP was initially assumed unsuitable for multimedia delivery [19]. This motivated considerable work to extend the capabilities of UDP for carrying video streams and coexisting with regular TCP traffic.

1) RTP and multicasting: RTP was proposed in RFC3550 [24] as a protocol on top of UDP for real-time streaming.UDP-based reliable data delivery and congestion control were not part of its original specifications. A great amount of work has been done to augment RTP with application layer congestion control for both unicast and multicast cases. Such techniques usually involve rate control, that is matching the rate of the video stream to the available bandwidth. However, both RTP congestion control and reliability remain open issues. Although video codecs are usually designed to deal with minor data loss, a packet loss in an I-frame or header of an I-frame can cause serious disruptions to the video.

With multicast it can be difficult to find a stream rate acceptable to multiple clients having different hardware capabilities and network resources. There are two proposed solutions: One is to rely on the Internet to provide network QoS capabilities (e.g. reserving resources, maintaining bounds on delay, jitter and loss), and the other is to use adaptive bitrate techniques to adjust the stream bitrate to the available bandwidth.

Multicast-based adaptive bitrate techniques [28] can be classified into three main categories: single stream approaches, replicated stream approaches, and layered stream approaches. In the single stream approach, a single video stream is transmitted to the multicast group and feedback is received from all clients participating in the group. In the replicated stream approach, the same video is replicated in multiple streams (each with a different bitrate/quality) and the client can join a stream that fits its capability. In the layered stream approach, the server sends the video stream in multiple layers and each client can then subscribe to a subset of layers that fits its processing power and network speed.

Some recent work has continued to build on augmenting RTP for video delivery, particularly in the Web Real-Time Communication (WebRTC) framework [29], RTP multimedia congestion control [30], [31] and multipath RTP [32], [33] mechanisms to regulate the transmission of video packets. Although multicast is implemented on most routers, Internet Service Providers (ISP) generally do not enable multicast on their networks. Because of these reasons, RTP on multicast did not provide an attractive platform for delivery multimedia over the Internet.

2) Peer-to-Peer (P2P) streaming: P2P networks [34] allow users to share content without the need for centralised servers, making it an attractive solution for delivering video over the Internet. There are two categories of P2P systems: tree-based and mesh-based. In tree-based systems, peers are organised in a tree structure and video is usually pushed from the root to subsequent levels of the tree until it reaches the leafs. Although this model is simple and easy to control, it can be severely affected by “peer churn” [35]. In mesh-based systems, peers connect to a random set of neighbouring peers who watch the same content. Peers usually exchange information about their data availability and then they retrieve data from neighbours when it is ready. Since each client maintains a set of peers at any point in time, this model is much less susceptible to peer churn than the tree-based model.

Multiple adaptive streaming techniques have been proposed in P2P streaming systems. Layered video encoding has been used to adaptively deliver different layers of the video the clients. Multiple Description Coding (MDC) and network coding has also been used to propose adaptive streaming systems that support a large number of users. P2P streaming in the form of BitTorrent is still popular for video file sharing today.

3) HTTP video streaming: In the early 2000s, researchers accepted that TCP offered some benefits for delay-tolerant video transmission. An application layer playout buffer was introduced to compensate for the rate fluctuations of TCP. Leveraging HTTP on top of TCP also proved to be very convenient, yielding several benefits (see Section III-E). The initial implementation of delivering video over HTTP/TCP is called Progressive Download – the client simply downloads the entire (one) video file (with constant video quality) as fast as TCP allows. The video player at the client-end starts video playback before the download is complete. One major drawback of this technique is that different clients with different capabilities across different network connections receive the same video quality, which can cause unwanted playback stalls. This led to the development of HTTP Adaptive Streaming (HAS) or DASH[3] in the mid-2000s. A DASH video client can adaptively request different video bitrates so that it matches the bandwidth that the network can support.

The key differences between DASH and earlier protocols for multimedia streaming are:

- Unlike earlier UDP-based schemes, DASH is built on top of TCP transport.
- The client drives the algorithm. Depending on its ABR, the client typically requests video bitrates based on observed network conditions, hence regulating the server's transmission rate.
- DASH requests and receives video data in terms of multi-second video chunks instead of a continuous stream of video packets.

Although various video streaming technologies mentioned above are still in use, the video streaming industry has now settled on DASH as the main component of video delivery over the Internet. We elaborate on current DASH architecture and ecosystem in Section III.

[3]Although the terms HAS and DASH can be used interchangeably, we will use DASH acronym from now on.

III. State-of-the-art: Dynamic Adaptive Streaming over HTTP (DASH)

This section presents an architectural overview of DASH and its applications, the benefits of using HTTP and the general principles driving the rate adaptation algorithms.

A. Architectural overview

In DASH systems [36] (summarised in Figure 1), video content is encoded into multiple versions at different discrete bitrates (*representation rates*). Each encoded video is then fragmented into small video segments or chunks, each containing a few seconds of video. Chunks from one bitrate are aligned in the video time line to chunks from other bitrates so that the client can smoothly switch bitrates, if necessary, at the chunk boundary. Content information such as video profiles, metadata, mimeType [37], [38], codecs, byte-ranges, server IP addresses, and download URLs is described in the associated Media Presentation Description (MPD) files. The MPD describes a piece of video content within a specific duration as a *Period*. In a Period, there are multiple versions of the content, each known as a *Representation*. In a Representation, there are multiple video *segments* or *chunks*. URLs pointing to the video chunks in an MPD can either be explicitly described or be constructed via a template (client deriving a valid URL for each chunk at a certain Representation) [36]. Video chunks are 3GP-formatted [37], [38] and in each Representation, there is a single initialisation segment which contains the configuration data and many media segments. Concatenating the initialisation segment and a series of media segments results in a continuous stream of video. Video chunks and MPDs are then served to clients by using standard HTTP servers.

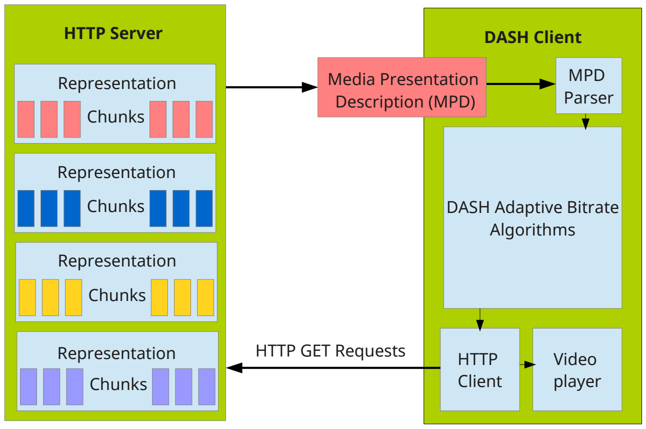

Fig. 1. DASH client-server architecture

Unlike traditional streaming strategies, DASH does not control the video transmission rate directly. It depends on the underlying TCP algorithm to regulate the video transmission rate, which is determined by the congestion feedback from the client-server network path. When a streaming session starts, the client requests the MPD file from the HTTP server and then starts requesting video chunks (typically in sequential order) as fast as possible to fill the playout buffer. Once this buffer is full, the player enters a steady state phase where it periodically downloads new chunks according to its chosen ABR algorithm.

In the steady state, the player is in the ON state when it is downloading a chunk, and in the OFF state otherwise (resulting in an alternating ON-OFF traffic pattern illustrated in Figure 2). The time between the start of two consecutive ON periods is termed cycle time (typically the chunk size – amount of multimedia content within each chunk – in seconds). The client typically keeps a few chunks in the buffer to maintain adequate playback.

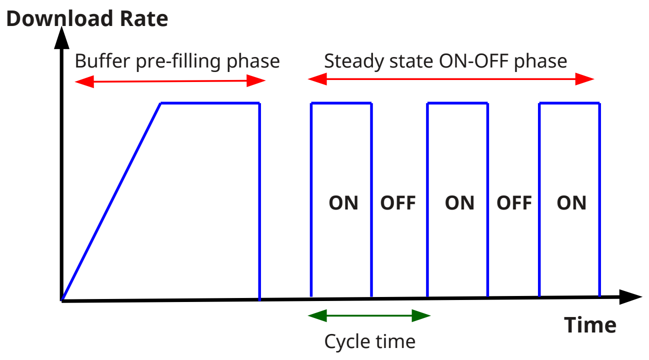

Fig. 2. DASH's bursty ON-OFF behaviour

The video player uses various feedback signals observed for each chunk (such as recently achieved throughput[4] and/or playout buffer occupancy) to select a suitable video rate for the next chunk to be downloaded. Consider an example when using achieved throughput as a criteria. If the throughput is high, ABR should select a higher video rate to provide better QoE for the user. On the other hand, if the throughput is low, ABR should dynamically switch to a lower video rate to avoid playout buffer under-run. A good ABR algorithm is responsive to fluctuating network conditions and adapts smoothly to provide better QoE [39].

The presented video bitrate (or quality) is limited by the video rates provided by the server, the information contained in the MPD and the network bandwidth. The DASH clients cannot match the network throughput perfectly; they can only achieve the (discrete) video rates described by the MPD. It will select a rate below the estimated throughput to sustain video playback and in the case where the network bandwidth exceeds the maximum video bitrate, the video rate is capped to the maximum video bitrate. Hence, in some ABR approaches, the server can artificially limit the video rate by only providing specific rates in the MPD to protect the network. The "smoothness" between video bitrate transitions depend on the encoding granularity (the number of video representations) of the video content provided at the server.

[4]Estimated from the size of previous chunks and the time taken to retrieve them.

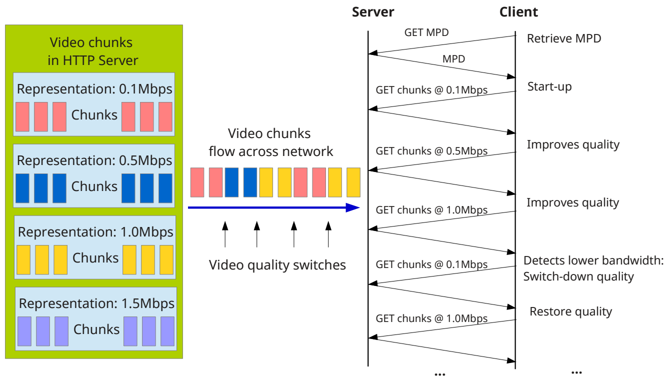

Fig. 3. Timeline illustration of DASH adaptive behaviour

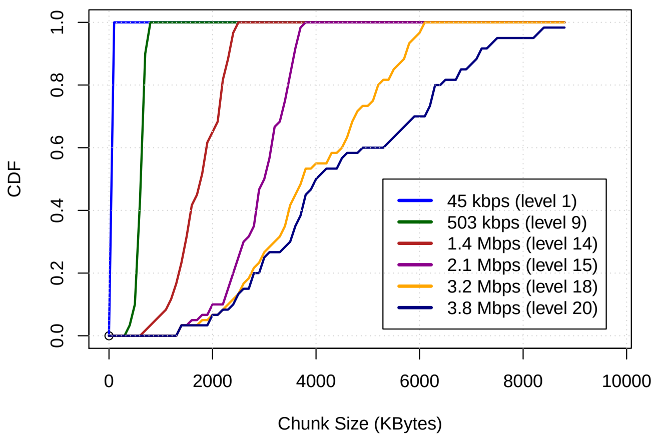

Fig. 4. Size distributions of 10-sec video chunks for six representation rates

As illustrated in Figure 3, the client quickly ramps up its video bitrate request during start-up to pre-fill its playout buffer. When the client detects a reduction in bandwidth capacity (by utilising feedback signals from previous chunks), it "backs off" by requesting a lower video bitrate. When the network capacity increases again, it then restores its video quality. As a result, the client is able to stream the video seamlessly without having to over-provision the network or keep an oversize playout buffer.

We illustrate the relationship between video chunk sizes and representation rates using the publicly available "Big Buck Bunny" dataset [40] (an open-source, 9mins 46secs long animated video by the Blender Institute).[5] Table I shows the resolutions and representation rates of the dataset's 20 encoding levels when chunks are encoded with 10 seconds of video. Figure 4 shows the chunk size distributions (in kilobytes) of six different encoding levels from Table I; video encoded at higher bitrates results in larger chunk sizes for the same 10 seconds of video contained within them.

TABLE I
REPRESENTATION RATES FOR 10-SEC DATASET

Resolution	Encoding Level	Representation Rates
320 x 240	1 - 3	45, 88, 127 kbps
480 x 360	4 - 8	177, 217, 253, 317, 369 kbps
854 x 480	9 - 10	503, 569 kbps
1280 x 720	11 - 14	0.8, 1.0, 1.2, 1.4 Mbps
1920 x 1080	15 - 20	2.1, 2.4, 2.9, 3.2, 3.5, 3.8 Mbps

[5]Full-length video sequences encoded at different bitrates, resolutions and chunk sizes can be downloaded from https://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny

B. Live Streaming

Live streaming of events has been traditionally done by using broadcast TV. DASH can also be applied to live streaming over the Internet [41], despite the tighter latency constraints compared to on-demand streaming services. The challenge of live streaming is to minimise the time (end-to-end delay) between the content generation (at the server) and presentation (at the client), which translates into the "liveliness" of the video stream. Broadcast TV aim to minimise end-to-end delays by continuously sending video content to the client. However, in the DASH architecture, retrieval of segmented/chunked video content is driven by the client. The processes of encoding, segmentation and transport video in response to the client's requests at the server-end add extra delays.

The main difference between on-demand streaming and live streaming is the content generation time. In on-demand streaming, all content at the server has been generated in advance before streaming to the clients, whereas media content is generated on-the-fly in the case of live streaming. DASH may be used in both scenarios, but with key differences in MPD fields [42]. The *minimumUpdatePeriodMPD* field instructs the client to check for MPD updates for every given interval when new live content is available (this interval is usually few chunk-duration long). The *availabilityStartTime* field associates a global UTC (Coordinated Universal Time) time with the start time for the first period of in the MPD and *availabilityEndTime* field describes the time when the live event ends. The *minBufferTime* field ensures that the client keeps at least a certain duration of content in the playout buffer. The client will need to be aware of the "live-point" and select the appropriate period by frequently updating the MPD if the *minimumUpdatePeriodMPD* field is specified. This technique is also used for inserting advertisements and commercials during live sessions. DASH standard [36], [13] does not require frequent MPD updates for live sessions, so the MPD can describe future video chunks *and* the associated URLs, the client will then need to determine the latest available video chunk using the current time and *availabilityStartTime* field.

Client-side ABR algorithms used in on-demand streaming can be generally applied to live streaming [43]. There are methods in controlling the chunk duration [44] and HTTP request strategies [45] in order to maintain a high liveliness and seamless playback of live content.

C. Mobile/Cellular streaming

Mobile video traffic represent a growing fraction of total mobile traffic, accounting for more than half of total mobile traffic in 2015, and is expected to reach 75% in 2020 [46]. Streaming multimedia using DASH over mobile/cellular networks to mobile devices (e.g. smartphones, tablets, laptops) presents additional challenges due to limited cellular coverage, unpredictability of the path characteristics leading to high latency, and the heterogeneity of mobile devices (e.g. various CPU power, screen resolutions). Highly variable bandwidth and limited resources in mobile networks compared to fixed networks mean that TCP's behaviour can be different, which in turn affects the ABR selection process. Mobile networks typically have higher latency, more congestion and higher packet loss rates when compared with wired networks, leading to more TCP retransmissions which in turn degrades the timeliness of video packet arrivals. [47] provides an overview of the mobile content delivery architecture and discusses various methods, including traditional RTSP/RTP, progressive HTTP download, paced HTTP download, HTTP-based adaptive streaming used in delivery on-demand and live video content to mobile devices.

In [48] and [49] the authors analysed the behaviour of mobile clients when streaming on-demand and live video from an ISP perspective. They also analysed the video bitrate switching frequency and caching performance in mobile networks. They determined that the Log-normal distribution is best used to describe the number of chunks requested for both on-demand and live sessions for the first 40 chunks (this distribution is also used to described various mobile communication patterns, such as call holding times and mobile file transfer). When the stream exceed 40 chunks, the Generalised Pareto Distribution best models the behaviour.

Mobile DASH ABR algorithms are broadly the same, with some works proposing various enhancements to the client and cellular access points [50], [51], [52], [53], [54].

D. Standardisation

Move Networks (now acquired by Echostar) was an early proponent of HTTP-based adaptive streaming technology and was awarded a fundamental patent [55] on adaptive streaming by the United States Patent and Trademark Office (USPTO) in 2010. The patent covers the invention of video streaming over packet-switched networks, particularly the structure of video content and the intelligent requests sent by clients which allows for adaptive rate-shifting over IP networks.

MPEG-DASH has been standardised by 3GPP [56] (first open as Work Item in January 2009 and finalised in March 2010) and became an ISO/IEC (International Electrotechnical Commission) standard for adaptive streaming in 2012 [13]. The standard defines guidelines for media presentation, segmentation, and a collection of standard XML formats for the manifest file (MPD). However, specific client implementation and rate adaptation techniques are not part of the standard [36]. Hence, commercial streaming services that use DASH implement their own proprietary techniques both for media representation and for client adaptation. DASH has subsequently being adopted by other standardised multimedia streaming systems such as IPTV (Open IPTV standard [57]) and Digital Video Broadcasting (DVB) [58] as a standard for transporting video over the Internet.

The DASH Industry Forum (DASH-IF) [59] (a group containing leading streaming companies) drives the adoption and research in modern adaptive streaming technologies. DASH-IF provides specific implementation guidelines and regular documentation of interoperability [60]. The community has also developed an open-source dash.js [61] reference player, which employs Media Source Extensions (MSEs) in a web/HTML5-based[6] video player for research and testing purposes. DASH-IF also provides and compiles a comprehensive list of publicly available test datasets [62], video players/clients [63], software for content preparation and validation [64].

E. Benefits of using HTTP

By using HTTP on top of TCP, DASH yields the following benefits:

- Clients use the standard HTTP protocol which provides more ubiquitous reach as HTTP traffic can traverse NATs and firewalls [65].
- DASH servers are regular commodity Web servers, which significantly reduces the operational costs and allow the deployment of caches to improve the performance and reduce the network load.
- A client requests each video chunk independently and maintains the playback session state, so servers do not need to track session state. Maintaining session state at the client means clients can retrieve video chunks from multiple servers with load-balancing and fault tolerance between commodity HTTP servers [66], [67].
- Relying on TCP reliability and inter-flow friendliness improves the likelihood that streaming traffic consumes only a fair fraction of the network bandwidth when sharing with other traffic.

These advantages enable service providers to leverage existing and significantly cheaper HTTP infrastructures. Proprietary commercial systems such as Microsoft's Smooth Streaming [16], Adobe's HTTP Dynamic Streaming (HDS) [17] or Apple's HTTP Live Streaming (HLS) [18] leverage existing CDNs and proxy caches.

F. Quality of Experience (QoE) metrics

Like other applications, multimedia content delivery over IP networks has to cope with inherent network QoS limitations. Bottlenecks can occur in various places such as at the server-end, ISP networks interconnection, the access link, the last-mile or within the home network. When traffic overloads these

[6]The web community is moving away from Adobe Flash towards pure HTML5. For example, starting from version 55, Google Chrome turns Flash player off and runs HTML5 by default on most sites.

bottlenecks, delays and packet losses occur, which can strongly impact the user's QoE. Since DASH is layered on top of TCP, its behaviour can be unpredictable due to TCP's dynamic behaviour, hence having various impact on QoE. Achieving high QoE is a major challenge due to the sheer diversity of video-capable devices (smartphones, tablets, desktops, smart-TV) and how they are connected to the network (cable, fibre, DSL, WiFi, cellular wireless), each providing different link-layer bandwidth characteristics.

With the advancement in DASH-based technologies, there is a noticeable shift from traditional QoE measurement on video quality (e.g. Peak Signal-to-Noise Ratio) and user experience (e.g. subjective mean opinion scores) to more complex quality metrics (e.g. rebuffering time, video bitrate, startup delays) and engagement-centric metrics (e.g. views per video, viewing duration). Optimising DASH's QoE is a challenging goal because these metrics are often interdependent, having counter-intuitive and complex relationships. Seufert et al. [68] provides a comprehensive survey of the available QoE metrics investigated in both industry and academic research, hence we only present some of the key works on QoE measurements here.

In one of the early works investigating QoE [69], Cranley et al. claimed that there exists an encoding which maximises QoE for a given target bitrate which can be extended to optimal adaptation trajectory for a whole video stream. They focused on the adaptation of MP4 video streams within a 2D space defined by frame rate and spatial resolution. Comparing clips with similar bitrates, they showed that reduction in frame rate is perceived as being worse than the reduction in resolution.

These implications can also be applied to DASH in terms of users' sensitivity to frequent bitrate switches. In [70] Mok et al. showed that DASH streams' initial delay and stalling or rebuffering are the key influencing factors of QoE. They also showed that the change in video quality introduces a new perceptual dimension. Dobrian et al. [71] quantified the impact of different video types and quality (live, long/short video-on-demand) on user engagement at a per-view and per-user level using Kendall correlation. They found that rebuffering has the largest impact on QoE across all types of content, particularly live content. Average bitrate was also found to play a significantly more important role in live content than in on-demand content.

Balachandran et al. [72] identified the challenges in converting low-level quality metrics (buffering, start-up time, bitrate, bitrate switches) into a unified QoE. Since these metrics are interrelated in complex counter-intuitive ways, their relationship can be unpredictable. The nature of the content can complicate the interactions as well. Hence, they believe there is a need to develop a unified (how the set of QoE metrics taken together impact quality) and quantitative (how much does changing one metric impacts user engagement) understanding of how low-level quality metrics impact measures of experience. They showed that the issue of inter-dependency can be addressed by using a machine learning approach to build a suitable predictive model of user engagement from empirical observations. The same authors published a follow-up paper [73] and presented a data-driven predictive model of Internet video QoE. This machine learning approach captures the interactions between various metrics as well as their confounding effects. They demonstrated significant practical benefits that can be reaped by content providers when using such a predictive model.

G. General ABR principles and goals

DASH uses ABR algorithms to select the appropriate video bitrates dynamically upon changing network conditions,which involves two control loops [19] as shown in Figure 5:

- The *inner* TCP congestion control loop reacts to network congestion and tries to match the sending rate with the rate sustainable by the network.
- The *outer* ABR selection loop reacts to the rates that TCP dictates and tries to match the video bitrate to the average observed TCP rate/throughput.

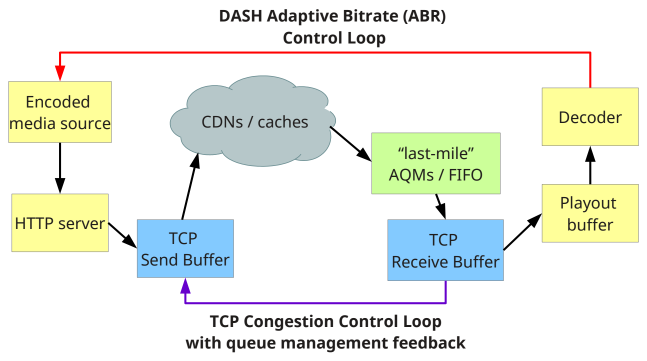

Fig. 5. Dual control loop with queue management feedback in DASH systems

The TCP control loop operates over the path's delay (typically in terms of milliseconds), whereas the ABR control loop operates over a larger time period (typically in terms of seconds). Typically, the ABR control loop does not try to match the short-term fluctuation of TCP throughput rather it tries to match a throughput averaged over a period of time.

The general goals of ABR [19] are:

1) Avoid playback interruptions caused by buffer under-runs.
2) Maximize the video quality (trade-off with goal 1 because it is always possible to minimize the number of interruptions by always transmitting at the lowest rate).
3) Minimize the number of video quality shifts to improve user experience (trade-off with goal 2 because the algorithm can maximize the video quality by reacting to the smallest changes in the network bandwidth, which, in turn, increases the number of quality shifts).
4) Minimize the time between the user making a request for a new video and the video actually starting to play (trade-off with goal 2 by using the lowest bitrate at the start).

The core challenge for ABR design is balancing these goals and ultimately providing a high QoE for end-users.

IV. Client-side Rate Adaptation Techniques

This section presents a survey of the research literature on *client-side* rate adaptation techniques.

Wang et al. [74] presented one of the most notable works in modelling multimedia streaming traffic over TCP in the early days of TCP-based streaming. This work forms the basis for other extended analysis and modelling described in [75], [76], [77]. The authors developed an analytic performance model to assess the performance of TCP when used to transport video streaming traffic without quality adaptation. Both theoretical and experimental results considered a constant bitrate encoded source, proved that TCP requires a network bandwidth that is roughly twice the video rate in order to achieve a good performance in terms of startup delay and percentage of late packet arrivals. However, this leads to wasting half of the bandwidth, prompting researchers to explore different adaptive streaming mechanisms to optimise both bandwidth usage and user's viewing experience.

Client-side ABR algorithms can be broadly classified into three categories based on the feedback signals they use: *throughput-based*, *buffer-based* and *hybrid/control theory-based*. We will look at examples of all three ABR approaches.

A. Throughput-based ABR

This class of algorithms rely on the TCP throughput (as estimated by the application layer) as the feedback signal for selecting subsequent video chunks. Throughput-based algorithms differ in terms of *how they estimate* throughput and *how they use* the estimates.

1) TCP-like AIMD based: Liu et al. [78] proposed a method that uses smoothed HTTP/TCP throughput measurement with TFRC/TCP-like AIMD (conservative step-wise up switching and aggressive down switching of representations) rate adaptation. This method does not require transport layer information such as packet loss rates and RTT. The method is capable of using smoothed throughput measurements to probe spare network capacity and detect congestion. It compares the chunk transfer (fetch) time with the media playback time contained in the chunk. The rationale is that a multi-second chunk is sufficient to smooth out short-term TCP throughput variations. A step-wise switch-up method (additive increase) is used to select higher representation for probing spare network capacity. An aggressive switch-down (multiplicative decrease) method is deployed upon detecting network congestion to prevent rebuffers. The algorithm also takes into account the idle time calculation to prevent client buffer overflow and consequently saving network bandwidth and memory resources.

The authors of [78] tested their methods using ns-2 [79] simulations and concluded that their results show the efficiency in detecting congestion and network capacity probing. They also showed that the retrieved video bitrate can quickly and accurately reach its optimum level.

2) FESTIVE: Jiang et al. [80] identified issues when multiple commercial DASH players share a bottleneck link with respect to three metrics: fairness, efficiency, and stability. The authors presented a principled understanding of bitrate adaptation and analysed several commercial players through the abstract player model consisting of bandwidth estimation, bitrate selection, and chunk scheduling components. They developed a suite of techniques that systematically guide the trade-offs between stability, fairness, and efficiency and this proposed a general framework for robust video adaptation.

Three potentially conflicting goals are identified in [80] that a robust ABR algorithm must strive to achieve (for N players sharing a bottleneck link with capacity W, with each player x playing/retrieving video bitrate $b_{x,t}$ at time t):

- (Un)Fairness: Multiple competing players sharing a bottleneck link should be able to converge to an equitable allocation of the network resources. Unfairness is defined as $\sqrt{1 - JainFair}$, where $JainFair$ is the Jain Fairness Index [81] of $b_{x,t}$ over all player x. A lower value means fairer allocation.
- (In)Efficiency: A group of players must choose the highest feasible set of bitrates to maximize the user experience. The inefficiency at time t is defined as $\frac{|\sum_x b_{x,t} - W|}{W}$, where a value close to zero implies that the players in aggregate are using as high an average bitrate as possible.
- (In)Stability: A player should avoid needless bit-rate switches as this can adversely affect the user experience. The weighted sum of all switch steps observed within the last $k = 20s$ divided by the weighted sum of the bitrates in the last $k = 20s$, the weight function $w(d) = k - d$ is used to add linear penalty to more recent bitrate switch. The instability metric is defined as
$$\frac{\sum_{d=0}^{k-1} |b_{x,t-d} - b_{x,t-d-1}| w(d)}{\sum_{d=1}^{k} b_{x,t-d} w(d)}$$

Unfairness, inefficiency and instability of players arise as a result of overlaying bitrate adaptation algorithms on top of several logical layers of network stack. Consequently, the feedback signal that the player receives from the network is not a true reflection of the network state. Furthermore, the feedback signal can be biased by the scheduling and bitrate selection decisions executed by the player. Periodic chunk scheduling combined with stateless bitrate selection was observed in [80] to create undesirable feedback loops with respect to the bandwidth estimation logic, causing unnecessary bitrate switches and potentially unfair allocation of bitrates.

At a high-level, the three-step process is: schedule when the next chunk will be downloaded, select a suitable chunk bitrate, and estimate the network bandwidth. The *harmonic bandwidth estimator* computes the harmonic mean of the throughput estimates for the last 20 seconds. The *stateful and delayed bitrate update* receives the throughput estimates from the bandwidth estimator and computes a reference bitrate. The *randomised scheduler* schedules the next chunk to be downloaded immediately if the playout buffer is less than the target buffer size. Otherwise, the next chunk is scheduled with a random delay by selecting a randomised target buffer size. The authors of [80] recommended randomised chunk scheduling to avoid synchronisation biases in sampling the network state; stateful bitrate selection that compensates for the biased interaction between bitrate and estimated bandwidth; a delayed update approach to trade-off stability and efficiency and a bandwidth estimator that uses the harmonic mean of the download speed over 20 recent chunks.

FESTIVE was implemented using Adobe's Open Source Media Framework (OSMF) [82] and compared it against several real and emulated commercial players across a range of scenarios that vary the bandwidth and number of users. FESTIVE was found to improve fairness by 40%, stability by 50% and efficiency by at least 10%. FESTIVE is robust to the number of players sharing a bottleneck, increase in bandwidth variability, and the available set of bitrates. FESTIVE is a client-side ABR with low implementation overhead and requires no modifications to the network or servers.

3) CS2P: Sun et al. [83] developed CS2P (Cross Session Stateful Predictor) to improve the bitrate selection and adaptation in video clients by using data-driven throughput prediction algorithms. They made three-fold contributions in their paper.

Firstly, they analysed the throughput characteristics of a large dataset provided by iQIYI, a commercial provider in China, which consists of 20 million sessions covering 3 million unique client IP addresses and 18 server IP addresses. They concluded that sessions sharing similar key features (e.g. ISP, geographical region) have similar *network* layer throughput values and dynamic patters, and there is a natural stateful behaviour in throughput variability within a session. This finding is similar to [84], where similar sessions have similar QoE performance at the *application* layer. However, there exists a complex relationship between session features and throughput. A client's perceived throughput is often influenced by multiple factors simultaneously, such as last-mile link technology, server load, backbone network congestion, which means that client sharing one of the features may not have similar throughput.

Secondly, the authors developed CS2P, a throughput prediction system that uses data-driven approach to learn clusters of similar sessions, predicts *initial* throughput, and models the stateful evolution of *midstream* throughput (intra-session) with a Hidden-Markov-Model.

Thirdly, they integrated CS2P in a dash.js client and showed with real-world deployment that it outperforms existing prediction approaches by 40% and 50% in terms of initial and midstream throughput prediction error. Besides, when combined with Model Predictive Control (MPC) [85] (see Section IV-C5) rate adaptation algorithms, CS2P has an overall improvement of 3.2% on QoE and 10.9% on average video bitrate over pure MPC with Harmonic Mean strategy. CS2P can also accurately predict the total rebuffering time at the start of the session. CS2P's session clustering approach is similar to that of [84] but it clusters sessions based on throughput prediction accuracy rather than application layer QoE metrics.

B. Buffer-based ABR

1) Buffer based rate selection: Huang et al. [86], [87], [88] are the first to propose, experimentally evaluate and field-test a pure buffer-based rate adaptation algorithm as illustrated in Figure 6. They demonstrated the challenges in estimating future capacity in commercial services caused by wide capacity fluctuations. Hence, they proposed an algorithm that uses only the buffer (by picking a video bitrate purely based on current buffer occupancy) during steady-state phase and probes for capacity estimation when the buffer is too low (not enough information for decision making, e.g in the start-up phase). They field-tested their algorithm in Netflix's clients and showed that this approach reduce the rebuffer rate by 10-20% compared to Netflix's then-default algorithm, while delivering a similar average video rate, and a higher video rate in steady state.

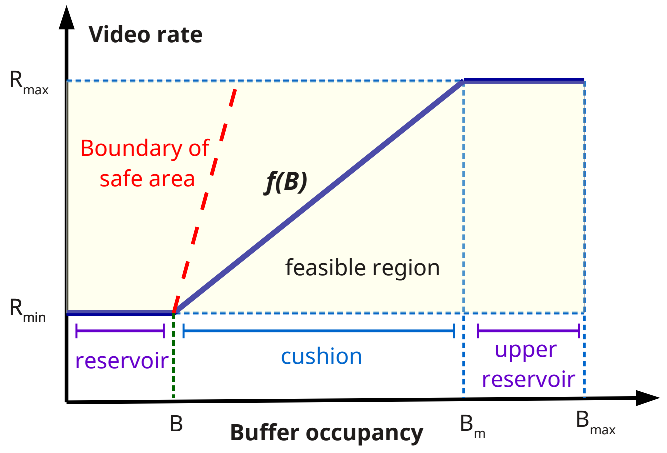

Fig. 6. Buffer-based adaptation rate map (adapted from Figure 6 in [86])

Figure 6 illustrates the approach's rate map – any curve $f(B)$ within the feasible region can produce a video rate between R_{min} and R_{max} given the current buffer occupancy. The lowest bitrate is selected when the buffer is filling reservoir B, after the reservoir is filled it selects the bitrate based on $f(B)$ algorithm. The buffer space between the reservoir and the point where $f(B)$ first reaches R_{max} is the cushion and the buffer after the cushion is the upper reservoir. The buffer has to be maintained above the reservoir so that there is enough buffer to absorb the variations caused by varying capacity. Hence, $f(B)$ will always download a chunk before the buffer shrinks into the reservoir area (operating in the safe area), otherwise it is in the risky area (boundary drawn in red dashed line).

2) Threshold based buffer : Miller et al. [89] proposed an algorithm that uses three threshold levels for the playout buffer, such that $0 < B_{min} < B_{low} < B_{high}$. The target interval B_{target} is between B_{low} and B_{high}, and the optimum interval $B_{optimum}$ is the centre of the target interval. The algorithm keeps the buffer level close to $B_{optimum}$. It allows the designer to explicitly control the trade-off between the variation in buffer occupancy and fluctuations in video bitrate in response to varying TCP throughput by controlling the B_{low} and B_{high} thresholds. They evaluated their prototype in real-world scenarios and found that it performs well under challenging situations. They ran simple flow with or without rate limiting experiments in a testbed and run real-world experiments in busy domestic WiFi environment. They observed that the algorithm managed to exhibit a stable and fair behaviour when multiple clients share a common network path.

3) Buffer Occupancy based Lyapunov Algorithm (BOLA): Spiteri et al. [90] formulated ABR as a *utility* maximisation problem and devised an online control algorithm, using Lyapunov optimisation techniques to minimise rebuffering and maximise video quality. BOLA achieved a time-average utility that is within an additive factor of the of the optimal value. This algorithm is also pure buffer-based algorithm which does not require any network bandwidth estimation and prediction.

BOLA uses a utility maximisation problem that incorporates both key QoE metrics – the average video bitrate and the rebuffering duration. An increase in the average bitrate increases utility, whereas rebuffering decreases it. The advantage of this framework is that utility can be defined in many ways, depending on the video content, provider and devices. This algorithm provides a *theoretical guarantee* on the achieved utility and provides an explicit knob for video providers to set the relative importance of a high video quality in relation to the probability of rebuffering. They provided the first theoretical justification for the reason buffer-based algorithms perform well in practice. However, BOLA assumes that the buffer level is at a sufficient level to provide all the information about past bandwidth variations.

The authors evaluated BOLA on 12 test vectors provided by DASH-IF with 85 publicly available 3G mobile bandwidth traces. They compared the results with an optimal offline algorithm that guarantee the maximum achievable time-average utility for any given network trace (having the perfect knowledge of future bandwidth variations) and found that BOLA achieve within 84-95% of the offline optimal utility in all test cases. The authors also compared BOLA with ELASTIC and PANDA (presented in Section IV-C), and concluded that BOLA is superior in terms of higher utility and consistency.

BOLA is now part of an experimental algorithm integrated in dash.js [61] since version 2.0.0.

4) ABMA+: Beben et al. [91] proposed an enhanced ABMA+ (Adaptation and Buffer Management Algorithm), which selects video representation rates based on the predicted probability of video freezing (inferred from the buffer occupancy). It continuously estimates segment download time characteristics and uses the pre-computed playout buffer map to select the maximum video representation which guarantee smooth content playout. Using a pre-computed buffer map minimises computation costs and simplifies deployment on different terminals. The authors validated ABMA+ with simulations and experimental trials using VLC player's DASH plugin [92]. They also compared their approach with rate and other buffer-based approaches and confirmed that ABMA+ more efficiently adjust video representations to variable network conditions, minimising video freezing and preventing frequent video representation switches. The authors also proposed an improved model for DASH systems that use buffer maps that closely reflect the impact of adaptation control logic by using state dependent description of the arrival process.

C. Hybrid/Control Theory-based ABR

ABR heuristics based on control theory use both throughput estimates and buffer occupancy as indicators of network conditions and repeatedly (re)solve a control theory/stochastic optimal control equation for selecting the representation rate to satisfy users' QoE preferences.

1) Smooth Rate Adaptation: The work done in [39] and [93] aim to control the playout buffer occupancy, keeping it to a reference level and uses the difference between the current buffer level and reference level to drive the control loop. Then, the equations for the predicted video rate can be derived as a function of the buffer size difference and the estimated TCP throughput which in turns determines the buffer occupancy level in combination with the actual TCP throughput, thus closing the control loop.

Trying to keep the playout buffer at a certain reference level turns out to not be the most suitable way to drive the control loop. From the users' point of view, it is more important to receive the highest (and most consistent) video quality sustainable while reducing bitrate fluctuations. The model described above does not take into account the bitrate fluctuations.

2) PANDA: Zhi et al. [94] used testbed experiments to demonstrate the fundamental limitations of relying solely on throughput estimates. They showed that when multiple DASH clients compete at a network bottleneck, the discrete nature of the video bitrates results in difficulty for a client to correctly perceive its fair-share bandwidth. This leads to video bitrate oscillations and low QoE. Hence, they proposed a Probe And Adapt (PANDA) mechanism for bitrate adaptation which is akin to TCP congestion control. PANDA probes the network by setting a target average data rate, then this data rate is subsequently used to determine the next video chunk bitrate and the subsequent request interval. Its AIMD probing mechanism is similar to TCP congestion control. The fundamental difference is that TCP congestion control detects congestion when there are packet losses (loss-based) or increases in RTT (delay-based), whereas PANDA detects congestion with the reduction of throughput. This property ensures that PANDA is able to efficiently utilise the network bandwidth, and in the presence of multiple clients, the bandwidth for each client eventually converges to a fair-share level. PANDA schedules the next request by considering the average target rate and buffer level. It presents a buffer filling based adaptation algorithm that solves the quality selection optimisation problem, and uses peak signal to noise ratio (PSNR) to capture QoE. Using testbed experiments, they showed that PANDA is able reduce video instability by over 75% when compared with other conventional algorithms, without rebuffers.

3) SQUAD: Wang et al. [95] proposed Spectrum-based Quality Adaptation for DASH (SQUAD) using both throughput and buffer feedback to develop a spectrum-based quality adaptation technique to ensure high QoE. The authors aim to maximise the average quality and minimise the number of quality changes. They use *"spectrum"* (a centralised measure for the variation of quality bitrates around the average quality) as a metric to capture QoE, and claimed that SQUAD solves the discrepancies of application layer bandwidth estimation and the underlying transport protocol by rate estimation. SQUAD was tested against multiple DASH players in a controlled testbed and in a cross-Atlantic Internet environment.

While trading off little to nothing in terms of average bitrate, SQUAD was seen to provide significantly better QoE in terms of quality switching frequency and magnitude (spectrum).

4) SABRE: Mansy et al. [96] demonstrated and quantified the *bufferbloat* [97] effect of DASH flows on other applications sharing the same bottleneck in a home network. Hence, they developed Smooth Adaptive Bit RatE (SABRE) in VLC player to mitigate this problem. They showed that SABRE can reduce queuing delays and significantly reduce the bufferbloat effect.

SABRE dynamically adjusts the TCP receive window (`rwnd`) in a DASH client from the application layer so that the burst size from the server to client is effectively reduced to the average queue size of the home router. The key techniques are: HTTP pipelining, controlling video chunk download rate and dual backoff or refill mode of operation.

HTTP pipelining is used to send multiple HTTP GET requests simultaneously to keep the client receive buffer always full, so it does not advertise a large `rwnd`. SABRE computes the number of HTTP requests the client should pipeline by using information on the actual size of the receive buffer, the chunk size and chunk bitrate. The algorithm controls the download rate at the application by computing the rate of the socket `recv` call so that the achieved download rate at any point is not much higher than the video bitrate. The rationale is to control the rate at which the application read from the receive buffer, hence controlling the growth of `rwnd`. Since, the there is no need to read data from the socket buffer at a rate higher than the video bitrate, SABRE distributes the `recv` calls uniformly over the chunk size (in secs). This maintains a steady download rate close to the target rate for a period of chunk size, and controlling the growth of `rwnd`. SABRE also uses the playout buffer occupancy for video bitrate switch-up or down, operating in a refill mode and a backoff mode. When the playout buffer level drops below a threshold, it operates in refill mode, once it exceeds another threshold, it enters the backoff mode to prevent overfilling the playout buffer.

The authors also ran experiments where multiple clients (SABRE and traditional players) compete for bandwidth and showed that SABRE can coexist well with other players without having performance penalties.

5) Model Predictive Control : Yin et al. [98], [85] proposed a Model Predictive Control (MPC) based algorithm that can optimally combine throughput and buffer occupancy feedback signals and formulated the rate selection problem as a stochastic optimal control problem. MPC attempts to predict key environment variables over a moving look-ahead horizon and solve an exact optimisation problem based on prediction. It can also explicitly handle complex and control objectives and constraints and has a set of well understood tuning parameters, such as prediction horizon. In this algorithm, MPC predicts the expected throughput for the next few chunks and using this to make optimal bitrate decision for QoE maximisation. They implement the variants of the algorithms – FastMPC, RobustMPC under dash.js, and validated their approach using realistic trace-driven emulations. The theory takes multiple constraints into account for deriving the optimal bitrate, and provides a framework for comparing the emulated or actual performance of algorithm with the "ideal" (best possible) performance of the algorithm.

They proposed a comprehensive QoE metric that is a *weighted combination* of the average video quality, the average quality variations, rebuffering time and the startup delay. The MPC model attempts to solve the maximisation problem for video QoE, whose key elements are defined as (R_k, B_k and C_k represents the video bitrate, buffer occupancy and bandwidth capacity for chunk K respectively):

- Average video quality – the average per-chunk quality over all chunks:
$$\frac{1}{k}\sum_{k=1}^{K} q(R_k)$$
- Average quality variations – the magnitude of changes in the quality from one chunk to another:
$$\frac{1}{K-1}\sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_{k)}|$$
- Rebuffer – rebuffering occurs when the download time is higher than the playout buffer level, hence the total rebuffer time is defined as:
$$\sum_{k=1}^{K}(\frac{d_k(R_k)}{C_k} - B_k)_+$$
- Startup delay – the time between user clicking the "play" button and actual video playback, defined as T_s.

The combined QoE metric of video chunk 1 through K is defined as:

$$QoE_1^K = \sum_{k=1}^{K} q(R_k) - \lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)| - \mu \sum_{k=1}^{K}(\frac{d_k(R_k)}{C_k} - B_k)_+ - \mu_s T_s$$

where users will assign different weights (using λ, μ, μ_s) based on which of the four components is the more important to them. Larger weighting parameters indicate a higher concern for that particular components.

MPC has an offline optimisation outside the client for different scenarios, and an *online* section which uses table lookups to point to the per-calculated solutions. Bandwidth prediction is used for chunk selection, but its performance very much depends on the accuracy of the prediction, which in turn relies on the exhaustive offline optimisation computation outside the client. The authors compared MPC with other algorithms (rate-based, buffer-based, default dash.js and FESTIVE) and confirm the advantages with negligible increase in computation and memory requirements (inside the client).

6) ELASTIC: As identified by [99] and [87], the DASH steady-state ON-OFF traffic pattern can cause unfair bandwidth utilisation, server bandwidth under-utilisation, frequent bitrate switches in the presence of other video streams or long-lived/greedy TCP flow. Hence, De Cicco et al. [100] proposed ELASTIC (fEedback Linearisation Adaptive STreamIng Controller), which use *feedback control theor*y that does not generate an ON-OFF traffic pattern.

Unlike conventional approaches that uses two controllers (one to throttle video bitrate, another to regulate buffer level), ELASTIC only uses one controller that computes the video bitrate level to drive the playout buffer to a fixed set-point. ELASTIC generates a traffic pattern that is identical to any long-lived TCP flow. Using a controlled testbed, the authors investigated if ELASTIC can fully utilise the bottleneck, fairly share the bottleneck and obtain a fair share when TCP greedy flows share the bottleneck with video flows. They concluded that ELASTIC achieves a very high fairness and is able to get the fair share when coexisting with TCP greedy flows.

The authors also compared ELASTIC with PANDA and FESTIVE in scenarios where multiple DASH flows are competing with and without long-lived TCP flows. They found that all three algorithms manage to achieve a fair share of bandwidth when there are only DASH streams, but when competing with greedy TCP flow, ELASTIC is able to achieve a fair share while providing high quality video with minimal bitrate switches.

V. Server-side, transport layer and network layer considerations

We now look at some existing server-side, transport layer and network-level solutions for optimising DASH-based multimedia streaming. Although most work has proposed adaptive bitrate selection on the client-end due to its ease of implementation and scalability, there has been some work on optimising server-side bitrate selection and congestion control.

We exclude server side optimisations based on changing the representation encoding schemes, such as moving from H.264/AVC (Advanced Video Coding) codecs to SVC (Scalable Video Coding) or HEVC (High Efficiency Video Coding) encoding in order to reduce storage requirements and improve caching efficiency. Instead we focus on the server-side application layer and transport layer optimisation to serve the pre-encoded video chunks stored in a regular web server, and look into alternative transport protocols and in-network solutions proposed in recent research literature.

A. Server-side application layer

1) Bitrate switching: Akhshabi et al. [101] proposed a pure server-based traffic shaping method to reduce video bitrate oscillations and instability due to multiple players competing for bottleneck capacity. The root cause is the ON-OFF activity pattern, which can (depending on the temporal overlap of players' ON-OFF period) lead to bandwidth overestimation, and oscillations between video bitrate levels. They developed a *shaping module* that limits the throughput for each chunk to the encoding rate of the chunk, so that the download duration will roughly equal to the chunk duration, reducing/eliminating the OFF period (as long as the available bandwidth is higher than the shaping rate).

Their approach aims to stabilise the players and allow the player to request the highest video bitrate that will not lead to oscillations. By experimentally evaluating their method with different scenarios (multiple shaped/unshaped players, competing with a persistent TCP transfer) the showed significantly reduced instability without a significant loss of bandwidth utilisation (aggregate throughput/available bandwidth).

Shaping introduces extra overhead, so it is only activated when the server detects oscillations. This method is client-independent, but assumes that the players receive most successive video chunks from the same server (which is not always true in real-world situations). In addition, if traffic shaping if turn on all the time, the server cannot detect the bandwidth variations. Hence the module is occasionally deactivated to allow the server to estimate the available bandwidth based on the TCP throughput of unshaped chunks.

De Cicco et al. [102] proposed a *control-theoretic server-side stream-switching technique* using a Quality Adaptation Controller (QAC). This method employs feedback control theory based on the client's feedback to serve a particular video bitrate. QAC utilises two controllers: a playout buffer level controller whose goal is to drive the buffer length to a target length; and a stream-switching logic that selects the appropriate video level to be streamed. They compared their prototype with Akamai's Adaptive HD video server and found that QAC is able to throttle the video quality to match the available bandwidth with a transient of less than 30secs, it fairly shares the available bandwidth in the presence of cross-traffic, and Akamai underutilises the available bandwidth due to conservative heuristics. Mueller et al. [103] proposed an adaptation algorithm implemented at the proxy level by using the concept of fairness regarding client cluster.

2) Video pacing: In [104] Alcock and Nelson revealed that YouTube uses an application flow control technique, called "block sending algorithm". Satoda et al. [105] introduced a server-side adaptive video pacing algorithm that delivers video data just-in-time, known as "Zippy pacing". The technique delays the delivery of a chunk after the video server finishes sending the previous chunk. Firstly, the video server sends video chunks with no delay until the playout buffer reaches a sufficient amount of data. Subsequently, Zippy pacing calculates the pacing delay. It tries to keep the playout buffer size as close as possible to a configured value by controlling the sending of video chunks. This method addresses the problem of bandwidth wastage caused by unnecessary data download (users do not finish watching the video stream all the time). The algorithm processes the characteristics of predicting the future stochastic diffusion of TCP throughput. Instead of predicting a decisive value, their method predicts a stochastic throughput diffusion by using a Brownian motion model as the stochastic process model. It controls the target playout buffer size while taking into account future throughput. This method allows network operators to manage their bandwidth better for video traffic while maintaining QoE. The authors conducted experiments in both High Speed Downlink Packet Access (HSDPA) and Long Term Evolution (LTE) environments and showed that the method can decrease the playout buffer size by up to 42%.

B. Server-side transport layer

1) Impact of TCP under DASH: To understand the need for server-side transport layer modifications, we first present the impact of TCP on DASH traffic. The timeliness (and therefore the rate) of chunk delivery can vary widely depending on the bottleneck network bandwidth, RTT delays, queue management and other cross-traffic competing for the path capacity. The use of TCP also means a DASH stream may cause collateral damage to other traffic sharing a network layer bottleneck [96].

The DASH server's TCP stack maintains a running estimate (the congestion window, `cwnd`) of how many bytes can be 'in-flight' and unacknowledged at any given time. `cwnd` starts low[7], grows as packets are received and acknowledged by the client, and shrinks when packets are lost or the connection goes idle for too long [106]. A DASH client's chunk retrieval process means TCP sends repeated bursts of packets followed by idle periods. If `cwnd` is too low, or fails to grow quickly enough, the DASH client experiences low chunk-throughput and if `cwnd` exceeds the path's bandwidth-delay product (BDP), the it will start filling bottleneck queues and induces queuing delays on other flows.

Huang et al. [87] analysed the interactions between TCP and DASH traffic by analysing traffic generated by major streaming services, and identified a vicious cycle they termed a *"downward spiral"*. This phenomenon is due to the fact that TCP resets `cwnd` to its initial value due to inactivity longer than the current retransmit timeout (around 200ms in their experiments) during the OFF period. Consequently `cwnd` ramps up in slow start mode to retrieve each new chunk. With no competing flows, the client still selects the highest sustainable video bitrate. However, when competing flows are present, the video flow is inflicted by high packet losses and low throughput, causing the client to select a lower video bitrate (smaller video chunks). With smaller chunks, TCP has less time to reach its fair share before it finishes each chunk download, leading to further underestimation of available bandwidth, leading to a downward spiral.

2) TCP modifications to avoid bursts: Much work has been done to model and understand how application rate-limited traffic affects TCP behaviour. Recently, a revised proposal on congestion window validation (CWV) has been submitted to IETF (RFC7661 [107]). The authors of [108] evaluated the effects of TCP CWV and TCP pacing on DASH transport.

Ghobadi et al. [109] proposed a server-side mechanism that uses TCP to rate-limit video traffic, which they called Trickle. To address the problem of TCP burstiness when delivering video which caused congested queues and packet losses, they designed Trickle to pace the video stream by placing an upper bound on TCP congestion window as a function of the streaming rate and the RTT. The server computes the `cwnd` bound from the observed RTT and the target streaming rate, and uses socket option to apply to the TCP. They evaluated Trickle on YouTube production data centres in Europe and India and analysed its impact on packet losses, bandwidth, RTT and video rebuffering events. The results show that Trickle reduces the average TCP loss rates by up to 43% and the average RTT by up to 28% while maintaining the streaming rate requested by the application.

3) Multipath TCP (MPTCP): Many service providers are capable of offering a number of independent paths, intra and inter-domain between two nodes. Also, many end-hosts today have multiple network interfaces (such as cellular and wireless interfaces on mobile devices), and this may yield the possibility for two endpoints to communicate via multiple paths. These characteristics can be exploited for load-balancing and bandwidth aggregation. There are several transport protocols that have been developed to use multiple network paths, such as SCTP uses multiple interfaces for redundancy/fail-over purposes, Multipath TCP (MPTCP) [110] offers parallel usage of multiple paths for resource pooling. Although both protocols are designed to load balance bulk transfers, MPTCP is gaining interest in the video streaming research community.

Very little work has been done on running DASH over MPTCP. This is partly due to potential performance degradation of multimedia content when the underlying paths are heterogenous, as shown in some experiments [111] and also lack of support in middleboxes [112], [113]. In a recent work [114], Corbillion et al. exploited the interactions between the application (video) layer and transport layers for MPTCP (undergoing active research by IETF, such as those documented in RFC8095 [115]) to support video streaming. Hence, they introduced a cross-layer scheduler, which leverages information from both application and transport layers to re-order the transmission of data and prioritise the most significant part of the video. They evaluated the performance of the cross-layer scheduler with traces aggregated from real MPTCP sessions (on Ethernet, WiFi and cellular accesses). They showed that the cross-layer scheduler improves the achieved QoE (video bitrate retrieved by the client) but still has efficiency limitations. However, the authors do not consider the adaptive mechanisms implemented in DASH. They worked on the basis that the ABR has already selected video chunk representation for delivery over MPTCP.

Another recent work by Han et al. [116] filled in this gap by proposing Multipath DASH (MP-DASH), a multipath framework for DASH streaming with awareness of users' network interface preferences. The basic idea is to strategically schedule chunk delivery to satisfy user preferences, such as preferring WiFi over cellular connection. Instead of improving video streaming quality with MPTCP, the authors goal is to reduce the overall streaming costs (e.g metered cellular usage). MP-DASH is designed to work with a wide range of ABRs and has two components: the MP-DASH scheduler and video adapter. The scheduler is overlaid on top of the MPTCP stack. It takes the interface preference from the user and the video delivery deadline from the player, then intelligently decides the best chunk downloading strategy over multiple paths while satisfying user preference. The video adapter is an add-on that lies between DASH ABR and MP-DASH scheduler, informing the scheduler the chunk size and deadline of the requested chunk. The authors integrated MP-DASH adapters into both throughput-based and buffer-based ABRs. The authors conducted experiments with these prototypes at 33 locations and proved that MP-DASH can be effective in reducing cellular

[7]The historical default was two packets, but in recent years some operating systems now start TCP connections with a window of ten packets.

usage up to 99% and radio energy consumption up to 85% when compared with using off-the-shelf MPTCP in the Linux kernel.

Chen et al. [117] performed experimental measurements on different applications using single-path TCP, two-path MPTCP and four-path MPTCP. They studied the latency distribution, video prefetch size, block size and periodic retrieval time for Netflix and YouTube streaming using both Andriod and iOS devices and showed MPTCP can be reasonably used for video streaming. However, Jurca et al. [118] showed by simulations some inherent issues and impact on QoE when streaming over multiple paths. The authors in [119] proposed MSPlayer, a video player that uses multiple video sources and network paths through WiFi/LTE interfaces. This technique is at the application layer, hence no client or server-side kernel modifications are required. They tested the player in an experimental testbed (with constant bitrate, not DASH) through YouTube and showed that it reduces start-up delay and provides high quality video and robust transport in mobile scenarios.

In [120] Wu et al. investigated the problems of using multi-homed terminals to stream video on mobile devices in a hetereogenous wireless network. Although not directly relevant to the adaptive mechanisms used in DASH streaming, they shed light on the behaviour of mobile video over multiple communication paths (taking the path asymmetry and data retransmission mechanisms into account). They developed an analytical framework for modeling MPTCP-based video delivery and proposed ADMIT (quAlity-Driven MultIpath TCP) which uses a utility maximisation based Forward Error Correction (FEC) coding and rate allocation to achieve optimal quality for real-time streaming. ADMIT uses a rate allocation algorithm to select the appropriate access network and then uses a FEC coding adaptation scheme to balance the end-to-end delay and packet loss rate. Their experiment results show that ADMIT improves video quality in terms of PSNR, and the benefits are more obvious when the number of access networks increases.

4) TCP Hollywood and ossified transport network: McQuistin et al. [121], [122] proposed TCP Hollywood, an unordered, time-lined, TCP variant designed to support real-time multimedia traffic. Their analysis indicates that it increases the utility of the network in lossy conditions where the latency requirements is constrained (VoIP, video streaming). Their experiments show that TCP Hollywood is deployable on the Internet, successfully operating on all major fixed and mobile networks in the United Kingdom.

TCP Hollywood is compatible with standard TCP, but eliminates two sources of transport-induced latency, and provides reliability measures that suits multimedia streaming applications. It removes Head-of-Line (HoL) blocking at the receiver and delivers received data to the application immediately, regardless of their order, and relaxes reliability to respect time lines provided by the application, so only data that will arrive in time is retransmitted, otherwise retransmissions carry new data. Both elements reduces latency and introduces message-oriented semantics, allow TCP Hollywood to express inter-dependencies between transmitted packets.

Their implementation uses an intermediate logic layer between the application layer and the kernel. TCP stack is modified to support out-of-order delivery and can be enabled or disabled via socket options. The concept of inconsistent retransmissions is introduced: if the RTT estimator indicates that a packet will arrive too late to useful, or if the packet depends on the previously unsuccessful transmitted packet, then TCP Hollywood will exploit the retransmission slots to send new packets instead of retransmitting useless data. TCP preserve the sequence numbers to determine if retransmission is required. The authors also developed an analytical framework to model the retransmission value against the buffering and processing time of the data at the receiver-side. They showed that under a wide range of RTT values, standard TCP retransmissions will cause packets to arrive too late to be useful, hence they use this model to validate TCP Hollywood and expect it to handle retransmissions correctly.

C. In-network solutions

In [123] Houdaille and Gouache proposed a *traffic shaping mechanism* (bandwidth manager) that allow bandwidth arbitration at the residential home gateway to address the problem of bitrate instability and unfairness when concurrent video streams are present. The traffic shaper intercepts the manifest files, determines the desirable target bitrates for each stream then constrains the clients to stay within their limits. This is described as delivering optimal QoE for the maximum number of users. Implementing the bandwidth manager at the home gateway has the advantage of being able to see and control all traffic coming into the home, and bandwidth can be allocated according to device roles and characteristics.

The approach in [123] was validated through experimentation with Microsoft Smooth Streaming player, which captured the stochastic nature of competing streams, high bitrate variance, unfair bandwidth sharing. They showed that their method provides benefits in terms of stability (low bitrate switching), bandwidth sharing accuracy and convergence speed (time taken to reach a stable bitrate).

Mok et al. [124] proposed a QoE-aware DASH system (QDASH) by using a *bandwidth measurement proxy*. They also carried out subjective experiments under Adobe OSMF, Apache server on Linux Debian, and concluded that users prefer a gradual change in quality when switching up or down.

QDASH consists of two modules – `QDASH-abw` and `QDASH-qoe`. `QDASH-abw` is implemented in a measurement proxy that is placed in front of the server and it probes and detect the highest quality level the current network conditions can support. It manipulates the video data packets to perform inline measurement by coupling the measurement flow with the video data flow. It measures the available bandwidth by using RTT estimates. On the client's end, `QDASH-qoe` helps the client to select the most suitable quality level receiving updates on the measurement results measured by `QDASH-abw`.

Proxy caching of multimedia streams substantially reduces the load on the network and server, reducing congestion in bottlenecks and video quality. In one of the earlier works of proxy caching for multimedia, Rejaie et al. [125] proposed a quality adaptive multimedia proxy cache for layered encoded streams, called MOCHA. The proxy can adjust the quality of cached streams based on their popularity and the available bandwidth between the proxy and clients. Hence, MOCHA can improve caching efficiency without compromising delivered quality. The algorithm implements fine-grained replacement and fine-grained pre-fetching mechanisms to adaptively increase or decrease the quality of cached streams. Although MOCHA is oriented towards RTSP for signalling and RTP/RTCP for streaming, the concepts can possibly be applied to DASH.

Pu et al. [126] proposed a proxy for video adaptation between fixed and wireless networks to increase the fairness for wireless clients. Mansy et al. [127] evaluated the performance of DASH streaming to mobile devices with different operating systems. They observed that unfairness can result when different device platforms are used. Siekkinen et al. [128] demonstrated that the bursty nature of DASH streams can be used for the benefit of power consumption in wireless networks. Havey et al. [129] proposed a receiver-driven rate-adaptive algorithm for wireless streaming.

D. Active Queue Management (AQM)

Evaluations are only beginning to emerge of DASH-based streaming over recently-developed AQM schemes. As illustrated in Figure 5, DASH can also be affected indirectly by the feedback signals from AQMs for regulating bottleneck congestion and fairness. The challenge is to understand the interactions between DASH's bursty traffic and the small effective bottleneck buffer emulated by AQMs.

Kua et al. [130] experimentally evaluated and characterised DASH-based content delivery over PIE (Proportional Integral controller Enhanced [131]), FQ-CoDel (FlowQueue Controlled Delay [132]) and FQ-PIE (FlowQueue PIE [133]) AQMs in a home network environment. They found that PIE's higher burst tolerance and queuing delay targets benefits a single DASH stream when there are no competing flows. However, in the presence of bulk cross-traffic, the FlowQueue Deficit Round Robin scheduler provides flow isolation and fair capacity sharing, hence protecting DASH flows from collateral damage in both upstream and downstream cases. They also evaluated a hybrid AQM, FQ-PIE which combines PIE individual queue management and FlowQueue scheduling algorithm, and showed that it provides the best results when DASH is competing with bulk TCP traffic for bottleneck capacity.

In [134] Gongbing et al. evaluated the performance of various traffic workloads, including DASH streams, bulk file transfer, Voice over IP (VoIP) flow and web traffic, using ns-2 to model a variety of AQMs (FIFO, Adaptive RED, CoDel and PIE) in a downstream DOCSIS 3.0 environment. They found that when five DASH flows compete with a varied number of bulk transfer flows, AQM significantly improves DASH performance compared to traditional FIFO schemes on the video bitrate. The results further suggest that CoDel leads to higher DASH adaptation rates. They also observed that PIE maintains the target queue latency more reliably than CoDel, and concluded that differences in the burst tolerances of PIE and CoDel contribute to observed differences in client adaptation rate.

E. Server and network assisted DASH

Due to the heterogeneity of devices accessing content and optimisation of QoE respectively, exchange of information between devices can be useful. The exchanged information can be leveraged by a video control plane enforcing network-assisted streaming strategies. A proposal on Server And Network Assisted DASH (SAND) [135] has emerged in the IETF to support active cooperation between network elements.

This technique enables a bi-directional messaging plane between clients and other DASH-aware Network Elements (DANE), allowing them to trigger a control mechanism such as flow prioritisation, bandwidth reservation and video quality adaptation based on the network state and clients. Network elements exchange PER (parameters for enhancing reception) and PED (parameters for enhancing delivery) messages. Software Define Networking (SDN) [136] is a viable technology to implement such mechanisms due to the presence of a centralised control element.

In [137] Kleinrouweler et al. proposed a DASH-aware networking architecture based on SDN. Network controllers with a broad overview on the network activity provide two mechanisms for *explicit adaptation assistance*: signaling target bitrates to DASH players and dynamic traffic control in the network to provide dynamic QoS. The controllers (at the control plane) ensure that the DASH players (at the data plane) can reach sufficient download speeds via QoS mechanisms, also assisting the players to select optimal bitrates. The authors evaluated their prototype in a WiFi setting and showed that the optimal video bitrate can be doubled and the number of quality switches are greatly reduced using their method.

This approach enables ISPs and network administrators to configure and define how bandwidth should be shared between video and non-video traffic, and how it should be shared among video players. Although explicit adaptation assistance enables stable streaming and fair sharing of network resource between DASH players, they will still under-perform when faced with long-lived cross-traffic. In order to reduce video oscillations in these scenarios, explicit adaptation assistance should be combined with QoS support. DASH players then can stream stably at a optimal quality level with or without cross traffic.

Cofano et al. [138] investigated several network-assisted streaming strategies (in SDN) which rely on active cooperation between video streaming applications and the network. They built a Video Control Plane which enforces video quality fairness among concurrent video flows generated by heterogenous client devices. A *max-min* fairness optimisation problem is solved at run time. They compared two approaches to actuate the optimal solution in an SDN network: *bandwidth allocation to video flows* and *video bitrate guidance*. The QoE metrics used are video quality, switching frequency and fairness. They also investigated the impact of different client-side ABR. They found that bitrate guidance provides the best results in terms of video quality fairness, whereas bandwidth allocation improves the average video quality depending on the ABR. ELASTIC and conventional (simple) rate-based algorithm were shown to provide higher video quality than PANDA, but simple rate-based ABR is affected by a large switching frequency. They also demonstrated that video fairness is improved when per-flow queuing is used within the network.

F. Predictive systems for server selection

Most video service providers today allow clients to switch CDN and bitrate for load-balancing purposes [66]. Hence, a global coordination platform or a logically centralised control system is required to assist with such decisions.

Jiang et al. [84] proposed Critical Feature Analytics (CFA), a scalable predictive analytics system for improving the QoE for Internet video applications by identifying critical factors that affect video quality. This work is built on insights from prior work presented in [67]. The authors' ultimate goal is to "choose the best CDN and bitrate for a client by accurately predicting the video quality of each hypothetical choice of CDN and bitrate". First they identified the key challenges of building an accurate prediction system – the complex relationships between video quality and features (e.g. autonomous system number, CDN, player, geographical region, video content) of the observed sessions, and the rapid changes of video quality. Hence, they saw a need for a prediction model that is expressive enough to capture these high dimensional relationships and capable of doing near real-time quality predictions.

CFA is driven by three key domain-specific insights – video sessions with same feature values have similar quality; each session has a subset of critical features that ultimately determines its video quality; and these critical features tend to be persistent. These characteristics enable CFA to learn critical features for different sessions on a coarsely grained timescales and update quality predictions in real-time.

CFA prediction consists of three stages – *critical feature learning* which runs offline (every tens of minutes) and outputs a critical feature function key-value table; *quality estimation* (runs every tens of seconds) based on the critical features learned and outputs a quality function key-value table; *real-time query and response* on the arrival of each client (on a millisecond timescale) by looking up the recently pre-computed value function.

The authors tested CFA with a major content provider and used it to optimise 150,000 sessions per day. Their results show significant improvements in terms of video quality, such as 32% less buffering time and 12% higher bitrate than a random decision maker. They also showed that CFA outperforms other machine learning (ML) algorithms such as Naive Bayes, Decision Tree, k-Nearest Neighbour. In order to evaluate their model more extensively in terms of algorithm comparisons and number of quality metrics, they ran trace-driven simulations and shows that CFA outperforms the baseline algorithm by 15%-52% and the best prediction algorithms by 5%-17%.

However, CFA currently makes predictions based on information provided by the clients only. While clients generally provide accurate information on perceived QoE, more accurate predictions can be made if information from the servers, caches and network paths are taken into account.

G. DASH streaming with Google SPDY, HTTP/2 and QUIC

Google's Chrome platform and YouTube services have a significant impact on Internet traffic [139], so it is worthwhile considering the impact of Google's underlying transport technologies on multimedia streaming.

Although various improvements such as persistent HTTP connections and pipelining (at the HTTP and TCP layer) have shown significant performance gain between HTTP/1.0 and HTTP/1.1, the HoL blocking problem together with TCP's streaming inflexibility still persist. Google developed SDPY [140] (later becoming HTTP/2, in RFC7540 [141]) and Quick UDP Internet Connections (QUIC [142], in Internet Draft [143]) to address these problems and reduce web latencies.

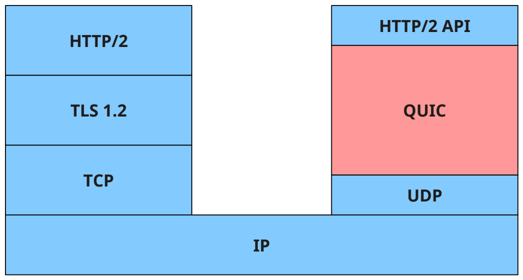

Fig. 7. HTTP/2 over TCP and QUIC: QUIC replaces part of HTTP/2, TLS and runs on top of UDP

SPDY was designed to be fully compatible with HTTP and could be integrated as a session layer between HTTP and TCP. SDPY multiplexes multiple streams on top of a single TCP connection per session. Network communication is based on frames that are exchanged between client and server. Figure 7 shows where QUIC fits, replacing part of HTTP/2 and TLS, providing a UDP-based transport for HTTP/2 with loss recovery and CUBIC-based congestion control [144].

A HTTP/1.1 client can only fetch one resource at a time (even with pipelining) whereas with QUIC a client can send multiple HTTP GET requests and receive multiple responses over the same UDP socket. HTTP/1.1 web browsers attempt to minimise the impact of HoL by opening multiple (typically six) concurrent HTTP connections. QUIC multiplexing features (inherited from SPDY) enables prioritisation among QUIC streams, traffic bundling over the same UDP connections and HTTP headers compression over the same connection.

In the context of DASH streaming, streaming over HTTP/1.1 introduces a latency of at least one chunk duration, which poses a problem for live video streaming. Wei and Swaminathan [145] developed a low latency live video streaming technique over HTTP/2.0 to address the problem. The authors employ a server-side method, known as the *server push* feature in HTTP/2.0 to stream live content actively from the server to the client as soon as the video chunks become available. They implemented this feature based on low latency mechanism, and showed that the method improves live streaming performance.

HTTP/2.0 specifies that the server push feature allows the web server to push a resource directly to the client without requiring the client requesting the resource. However, HTTP/2.0 is not designed for video streaming, so the authors realised that the server push mechanism cannot be directly adopted for video streaming unless a fully customised push strategy for live video is designed and evaluated. They evaluated various push strategies (No-Push, All-Push, k-Push) for live streaming. The evaluation results show that server push enables low latency live streaming by simply reducing the chunk size without increasing the request overhead.

Mueller et al. [146] presented one of the first works to compare DASH over HTTP/1.1 and pre-HTTP/2 (SPDY) with or without SSL encryption. They used VLC DASH player [147] to evaluate DASH's performance in terms of protocol overhead and performance over 0 to 150ms RTT. They found out that the overhead for all both HTTP versions are small, i.e 5-7% for 2-sec chunks and video bitrates higher than 700kbps. HTTP/1.0 achieves link utilisation equal or higher than 90% for RTT ranging from 0-50ms, but in the case of RTTs between 100-150ms, only 75-80% can be utilised.

Both HTTP/1.1 and SDPY perform consistently over all RTTs due to the persistent connection and pipelining features. Although both HTTP/1.1 and SPDY performs equally well, SPDY offers HTTP/1.1 functionalities implicitly. Despite the overhead introduced by SPDY framing and being not as efficient as HTTP/1.1, SPDY and SPDY with SSL encryption are very robust against increasing RTT because they are maintaining only one TCP connection during the whole session. They showed the SPDY implicitly solves the HoL problem and achieves good results when SSL is disabled.

However, SPDY mandates the encryption, which is unnecessary for streaming any multimedia content that is already Digital Rights Management (DRM) encrypted. Additional SSL encryption introduces additional computational overhead on both server and client.

Carlucci et al. [148] evaluated QUIC in terms of Web traffic and assessed the performance of QUIC compared to SPDY and HTTP/1.1. Preliminary experiment results and analysis on video streaming over QUIC are presented in bitmovin's reports [149], [150]. In the experiments, Timmerer et al. compared the performance of DASH streams when using HTTP/1.1/2.0/SPDY over TCP/QUIC using a controlled testbed. QUIC comes with a slightly higher protocol overhead than TCP but is below 10% except for very low bitrates ($\leq$100kbps). The link utilization decreases with increasing RTT but is always $>$ 87% of the available bandwidth and remains stable for different bandwidths.

Bottleneck Bandwidth and RTT (BBR) [151] is an emerging congestion-based transport protocol. BBR aims to maximise the network throughput with minimal queue by probing the bandwidth and RTT sequentially. BBR is now fully deployed for all TCP services on Google Wide Area Network (WAN) backbone, and has replaced TCP CUBIC on Google and YouTube services. BBR is now available as part of the Linux source tree in kernel version 4.9 and its implementation for QUIC and FreeBSD is currently under development. There are no documented experimental results on BBR's impact on adaptive video streams to date.

VI. Traffic Measurements and Characterisation

Several measurement studies have been conducted over the years to characterise traffic patterns and understand the technologies used by popular commercial streaming services (since most companies own proprietary technologies that are not revealed publicly). These studies attempt to understand DASH's adaptive behaviour "in the wild". In this section we discuss some of the key works carried out in measuring and characterising video traffic delivered with DASH technologies.

A. *Characterising traffic patterns*

To design and evaluate different rate adaptation schemes, it is necessary to understand the nature of the traffic the scheme has to deal with. The studies discussed in this section explore video traffic on long and short time scales. On long time scales, the patterns of popularity (the distribution of videos chosen) has attracted the most attention while on the short time scales the pattern of delivery during a streaming session has attracted most attention. These studies suggest that video popularity across large datasets appear to conform well to Zipf's law but for some smaller populations, such as a single University campus, there can be little correlation between what is globally popular and what is locally popular. Also, many videos were selected only once during the data collection period. On shorter time scales, there have been a number of strategies observed as to how video is delivered during a session. The dominant factor in strategy selection appears to be the client software.

In [152] Rao et al. did an in-depth analysis on the network characteristics of Netflix and YouTube traffic and showed that streaming strategies vary with the type of application (e.g. Web or mobile app) and container (e.g. Silverlight, Flash, HTML5). They identified three different strategies – no ON-OFF cycles, short ON-OFF cycles and long ON-OFF cycles, were used to stream commercial video and discussed how they impact the network. They used the amount of data transferred during the buffer prefilling phase, block size (amount of data transferred in one ON-OFF cycle) and accumulation ratio (ratio between the average download rate and the video bitrate during the ON-OFF steady-state phase) as their metrics for traffic characterisation. The authors observed that there are no ON-OFF cycles when Firefox is used to stream HTML5 videos, essentially resulting in a long-lived TCP flow. In addition, they observed that YouTube servers control the data transfer rate when any application uses Flash, resulting in short ON-OFF cycles, but do not explicitly rate limit when streaming HTML5 videos. Each application uses its own strategy to stream HTML5 videos. When streaming Netflix videos, web browsers and iOS devices use short ON-OFF cycles and Android devices use long ON-OFF cycles. In most cases, YouTube sends 40 seconds worth of video data during the buffer prefilling phase and serves 64 kB blocks during the steady state phase for Flash videos, resulting in short ON-OFF cycles. For HTML5 videos, Chrome and Android devices are observed to be downloading larger video blocks (>2.5 MB), with OFF periods in the order of 60 seconds, resulting in longer ON-OFF cycles.

The authors complemented their results by presenting an analytical model that evaluates the potential impact of these strategies on the aggregate traffic by considering video streaming parameters (e.g. video arrival rate, number of videos, encoding rate, video duration, buffering amount, accumulation ratio). This mathematical model provides insights for traffic engineering, for example, the potential impact of migrating from Flash to HTML5 containers.

Gill et al. [153] collected traces of 25 million transactions between an edge network (campus network) and YouTube, including 600,000 YouTube downloads over a three month period. They analysed and characterised YouTube traffic in the light of Web 2.0 (sites with user-generated content where users can upload content and view content posted by peers) from a local and global perspective. Locally, they consider the viewing habits of YouTube users on campus. Globally, they considered video popularity and examined their relationship with popular videos viewed on campus. They characterised YouTube video files in terms of file sizes, video durations, bitrates, age, ratings and categories. They subsequently examined the usage patterns, file properties, popularity, referencing characteristics, transfer behaviours and compared them to other traditional Web workloads. They observed that local viewing habits are well described by Zipf's law and concluded that video workloads are similar to Web workloads in terms of access patterns (closely correlated to human behaviours, e.g. traffic volumes are higher during a certain time, certain day of the week). The authors discussed the implications of Web 2.0 for network and service providers, and recommended the use of distributed CDNs and caches by leveraging the availability of metadata to reduce bandwidth consumption and core server workload.

Comparably, Zink et al. [154] did a similar study on the correlation between video bitrates, durations, global or local content popularity, and access patterns of YouTube videos by measuring streaming traffic in a campus network. They proposed a framework for monitoring YouTube signaling and data traffic by correlating TCP/IP and HTTP headers and packets exchanged at the campus gateway. They observed that there is no strong correlation between global and local popularity. They used data-driven simulations to demonstrate the advantages of using client-based local caching, P2P-based distribution and proxy caching in reducing video start-up times and reducing bandwidth consumption.

Similarly, Cha et al. [155] did an extensive data-driven analysis of YouTube and Daum (a popular streaming service in Korea). The authors studied the popularity life-cycle of videos (and what shapes the popularity distribution), the intrinsic statistical properties of requests and their relationship with video age, content duplication, the level of content aliasing or illegal content in the system. They applied Pareto and long tail analysis to identify the key elements that influence the popularity distribution. The authors concluded with insights into utilising peer-assisted techniques and caching to enhance user experience, which can potentially offload server traffic by up to 50%.

In [156] Finamore et al. compared the traffic patterns generated by mobile devices (smartphones, tablets) and PCs (desktops, laptops). They also investigated the users' behaviour and correlate it with the system performance. The authors performed these measurements by using unique datasets collected from various vantage points in nation-wide ISPs and university campuses in Europe and the United States. They discovered that user access patterns are broadly similar across a wide range of user locations, access technologies and user devices, with most users using default player configurations.They deduced that YouTube is highly optimised for PC users and less efficient when serving mobile devices. Together with limitations of mobile devices, YouTube serving strategies for these devices (more aggressive buffering) cause higher access time, lower download rate and more bursty traffic. Furthermore, they also discovered that 60% of the videos are watched for no more than 20% of their duration, resulting in waste of transferred data, which can impact mobile users more (due to limited storage and more expensive data usage), hence they proposed a more precise control of buffering in mobile devices with the use of efficient CDN caching schemes.

Most popularity predictive models focus on highly popular content – models that attempt to predict what content might be accessed next are, of necessity, based only on the most popular videos. Carlsson and Eager [157] took an orthogonal approach in analysing the implications of ephemeral content on content caching at edge networks. Their work is motivated by the fact that a large fraction of content is ephemeral (or "one-timers" – videos that are only requested once from the edge network), which means indiscriminate caching of all content at edge networks can lead to high inefficiency and costs. They performed an in-depth analysis on YouTube request characteristics observed at an edge network over a 20-month period. They observed that 71% of the videos are one-timers (requested only once in the 20-month period), hence demonstrating the need for selective caching policies. The authors then proposed a workload model for content delivery applications with ephemeral content. It models the distribution of total number of times that a content will be requested based on interests. The output is the used to analyse the trade-offs between cache insertion rate (content items inserted into the cache) and cache miss rate (requested content items that are not available in the cache). Subsequently, they applied their model to compare the performance of two edge caching policies – indiscriminate caching and cache on k^{th}request for different values of k. They find that caching based on the number of requests k can reduce cache insertions effectively. They also explored the possible benefits in terms of cache improvements based on other popularity prediction methods, such as the "oracle" policy.

B. Understanding commercial technologies

1) Reverse-engineering: Some studies have focused on understanding the operation of popular commercial streaming services by reverse-engineering the video streaming network architecture as a whole.

In [158] Adhikari et al. performed both active and passive measurements of Netflix to uncover its architecture and service strategy. They measured the available CDN bandwidth and investigated its behaviour at different geographical locations, and found that Netflix employs a blend of data centres and CDNs for content distribution. They also performed active measurements of the three Netflix CDNs to quantify the availability of video bandwidth. They proposed a measurement-based adaptive CDN selection strategy and a multiple-CDN-based video delivery strategy, and demonstrate their potential in increasing users' average bandwidth.

In [159] the authors deduced the key design features behind YouTube's content delivery system by using a distributed active measurement infrastructure (using multiple vantage points from PlanetLab nodes), analysing large volume of video playback logs, DNS mappings and latency data. They revealed the multiple DNS namespaces (reflecting a multi-layered logical organisation of video servers), a three-tier physical caching hierarchy, sophisticated mechanisms for handling cache misses (backend fetching) and load-balancing. In [160], Torres et al. used passively collected data to dissect server selection strategies for the same service and attempt to understand its load-balancing/multi-homing strategies.

The authors of [161], [162] studied the overall architecture of Hulu with active measurements. They focused on understanding Hulu's server selection strategies how resources are allocated to serve user requests, focusing primarily from the content provider perspective. They found out that Hulu CDN selection is done on control servers and communicated to clients through manifest files. Preferred CDNs are then assigned based on pre-determined probabilities, independent of location, video, time and instantaneous available bandwidth. All servers employ locality-based DNS resolution and customers from different services may share the same set of servers. Hulu frequently changes preferred CDNs and stay with the same CDN once the video is played (despite degradation of performance). They also observed that Hulu divides video requests among CDNs to attain a fixed target ratio. [163] presented the potential benefits of caching in Hulu's CDN. Netflix uses the same three CDNs that Hulu uses but assigns CDNs to user accounts statically. They also noted that Netflix uses DASH-like streaming whereas Hulu uses Real-Time Messaging Protocol (RTMP).

In [164] the authors presented an extensive trace-driven analysis of traffic exchanged between YouTube data centers and end-users, from the perspective of a large tier-1 ISP. They inferred the load balancing schemes and routing policies used to serve users' requests, and analysed their impact on traffic dynamics across YouTube and the ISP. They discovered that YouTube employs a location-agnostic, proportional load-balancing strategy among its data centres. They also proposed a novel method to estimate the unseen traffic (traffic carried outside of the ISP network). Their work help sheds light on the interplay between large content providers and ISPs.

2) Measuring from the client: Other studies focused on understanding the behaviour closer to the client-end.

In [87] Huang et al. measured how Hulu, Netflix and Vudu clients select video bitrates with the client-side bandwidth estimation. In all three services, the authors showed that inaccurate bandwidth estimates (especially in the face of competing bulk file download) can trigger a feedback loop that detrimentally lead to a variable and low-quality video, which they termed the *'downward spiral'* effect. The authors identified the root cause of the failure is the lack of information exchanged between TCP and HTTP. Hence, they proposed to design a client that enables TCP to reach a steady state fair share before reducing the video quality. A more radical solution is to eliminate the use of throughput estimates altogether, leaving rate control to TCP for attaining a fair share of bandwidth, and use buffer-occupancy to drive the video bitrate selection, resulting in the buffer-based rate adaptation algorithm presented in Section IV-B.

Liu et al. [165] did a comparative study between Android and iOS (on mobile devices) for accessing streaming services, using both server-side log analysis and client-side experiments. Android and iOS media players ares shown to use different content requesting approaches and different buffer management methods, resulting in different amounts of received data. iOS devices send out more HTTP requests than Android devices, and always use HTTP range requests, as opposed to the standard HTTP requests used by Android devices.

In [166] Akhshabi et al. experimentally evaluated the performance of Microsoft Smooth Streaming, Netflix and Adobe OSMF players in the scenarios with different variations of available bandwidths (unrestricted, persistent or short-term variations). They sought to understand how players react to persistent or short-term bandwidth variations, how quickly can a player converge to a sustainable bitrate, can two players share the available resources fairly and stably, and how does the player perform when streaming live content in terms of startup delay. They discovered that Smooth Streaming player is effective under unrestricted bandwidth as well as under persistent available bandwidth variations. It converges to the highest sustainable quickly but rather conservative in bitrate switching decisions. However, it reacts to short-term bandwidth availability too late, causing sudden drops in playout buffer and unnecessary bitrate reductions. Two competing Smooth Streaming players show that the logic is unable to avoid oscillations, and does not aim to reduce unfairness in bandwidth sharing. The Netflix player used in [166] also uses Smooth Streaming, hence their behaviours are similar. However, Netflix uses a much larger playout buffer and is more aggressive in providing the highest bitrates, even at the expense of additional bitrate changes. The OSMF player is unable to converge to an appropriate bitrate even after the available bandwidth has stabilised.

VII. Conclusions and Future Work

Due to its scalability and feasibility, DASH has emerged as a compelling standard for on-demand and live multimedia streaming over the Internet. The core essence of DASH is its ABR algorithms that enable the selection of appropriate video bitrates to match the dynamically changing network conditions. The DASH specifications provide flexibility for researchers and developers to explore and implement various

ABRs. Since network conditions are best known at the client-end, most ABRs focused on client-end heuristics. However, there are also other techniques that utilise server-end algorithms and network-level solutions to assist with clients' rate adaptation.

In this paper, we have surveyed key rate adaptation techniques and classified them in terms of the feedback signals used for video bitrate selection. Throughput-based ABR predicts the future network condition based on past chunk download rates. Pure buffer based algorithms use past and present buffer occupancy to determine the network state and choose a video bitrate that matches the network capacity. Most algorithms implemented are hybrid, combining both throughput and buffer as feedback signals for more accurate estimates. Control-theory based algorithms present the bitrate selection process and various QoE metrics as a stochastic optimal problem. It then attempt to solve the problem and drives the bitrate selection with a network bandwidth prediction horizon. Server-side transport and network layer modifications can also prove to be effective. Network-level solutions are proposed recently and will be feasible with the rapid developments in SDN and modern AQMs.

Throughput-based algorithms were thought to be sufficient for a smooth video viewing experience, but it turns out that sudden network fluctuations can be hard to predict. As user expectations increases, these algorithms start to take increasingly complex QoE metrics into account, using mathematical models to predict network characteristics and users' viewing behaviours. Pure buffer-based approaches are orthogonal, but do not account for user-defined QoE objectives. Hybrid client-side ABRs which combine both throughput and buffer-based signals, and subsequently apply them in a control theory framework, prove to be the most holistic approach. These algorithms smooth out throughput prediction errors and are able to maximise users' QoE preferences. A drawback of control-theory based ABR is that it can be computationally intensive. Although server and network-level solutions provide noteworthy benefits, they are significantly more complex and are less likely to be immediately deployable.

We have also covered some key traffic measurements and characterisation studies of notable commercial streaming companies. Since most companies own proprietary streaming technologies, these studies attempt to uncover the adaptive techniques used and the traffic patterns, behaviour and impact of DASH in the real world. Other studies attempt to understand the characteristics of DASH traffic when competing with various cross-traffic.

There remain several open research challenges and issues such as the following:

- Understanding the interactions between various classes of ABR algorithms and the different underlying TCP algorithms. For instance, experimentally analysing and characterising the impact of alternative transport protocols such as MPTCP, Google QUIC or BBR on DASH-based content delivery will be of great interest to the streaming community.
- Coupling of application and transport layer at the client-end so that DASH clients are aware of the underlying path's latency using transport layer RTT estimates.
- Design of client-side ABR algorithms that interacts optimally with modern bottleneck AQMs.
- Strategic placement of CDN servers and proxies.
- Server-side bandwidth management, resourcve allocation and pacing TCP packets to smooth out traffic burstiness.
- Fair resource sharing for DASH streams and other cross-traffic when multiple clients share a bottleneck.

The content streaming community has some interesting challenges ahead.

REFERENCES

[1] Sandvine, "Sandvine Global Internet Phenomena Report," 2015. [Online]. Available: https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, IETF, Jun. 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616

[3] J. Postel, "DoD Standard Transmission Control Protocol," RFC 793, IETF, Jan. 1980. [Online]. Available: https://tools.ietf.org/html/rfc793

[4] Cisco, "White paper: Cisco VNI Forecast and Methodology, 2015-2020." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[5] R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Internet Engineering Task Force, Sep. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4960

[6] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, Internet Engineering Task Force, Mar. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4340

[7] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631, Internet Engineering Task Force, May 1994. [Online]. Available: https://tools.ietf.org/html/rfc1631

[8] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Internet Engineering Task Force, Jan. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3022

[9] T. Hain, "Architectural Implications of NAT," RFC 2993, IETF, Nov. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2993

[10] B. Carpenter, "Internet Transparency," RFC 2775, IETF, Feb. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2775

[11] B. Aboba and E. Davies, "Reflections on Internet Transparency," RFC 4924, IETF, Jul. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4924

[12] J. Postel, "User Datagram Protocol," RFC 768, IETF, Aug. 1980. [Online]. Available: https://tools.ietf.org/html/rfc768

[13] ISO/IEC, "ISO/IEC 2309-1:2012 Information Technology: Dynamic Adaptive Streaming over HTTP (DASH) Part 1: Media presentation description and segment formats," 2012. [Online]. Available: https://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623

[14] LAME, "GPSYCHO - A LGPL'd Psycho-Acoustic Model," Oct. 2011. [Online]. Available: http://lame.sourceforge.net/

[15] G. Armitage, *Quality of Service in IP Networks: Foundations for a Multi-service Internet*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc., 2000.

[16] Microsoft, "Microsoft Silverlight Smooth Streaming," 2016. [Online]. Available: https://www.microsoft.com/silverlight/smoothstreaming/

[17] Adobe, "Adobe HTTP Dynamic Streaming (HDS)," 2016. [Online]. Available: https://www.adobe.com/devnet/hds.html

[18] Apple, "Apple HTTP Live Streaming," 2016. [Online]. Available: https://developer.apple.com/streaming/

[19] S. Varma, "Flow Control in Video Applications," in *Internet Congestion Control*. Morgan Kaufmann, 2015, ch. 6, pp. 173–202.

[20] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Mar 1999, pp. 1337–1345 vol.3.

[21] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 5348, IETF, Sep. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5348

[22] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," RFC 2205, Internet Engineering Task Force, Sep. 1997. [Online]. Available: https://tools.ietf.org/html/rfc2205

[23] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks," RFC 2998, Internet Engineering Task Force, Nov. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2998

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, IETF, Jul. 2003. [Online]. Available: https://tools.ietf.org/html/rfc3550

[25] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, IETF, Apr 1998. [Online]. Available: https://tools.ietf.org/html/rfc2326

[26] M. Handley and V. Jacobson, "SDP: Session Description Protocol," RFC 2327, IETF, Apr 1998. [Online]. Available: https://tools.ietf.org/html/rfc2327

[27] T. Friedman, R. Caceres, and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)," RFC 3611, Internet Engineering Task Force, Nov. 2003. [Online]. Available: https://tools.ietf.org/html/rfc3611

[28] CableLabs, "IP Multicast Adaptive Bit Rate," Cable Television Laboratories, Inc., Tech. Rep. OC-TR-IP-MULTI-ARCH-C01-161026, 26 October 2016. [Online]. Available: https://community.cablelabs.com/wiki/plugins/servlet/cablelabs/alfresco/download?id=3edb1609-17ff-4844-87ed-124314a73e7c

[29] C. Holmberg, S. Hakansson, and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements," RFC 7478, Internet Engineering Task Force, Mar. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7478

[30] C. Perkins and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions," RFC 8083, Mar. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8083

[31] V. Singh, J. Ott, and S. Holmer, "Evaluating Congestion Control for Interactive Real-time Media," Internet Engineering Task Force, Internet-Draft draft-ietf-rmcat-eval-criteria-06, Sep. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rmcat-eval-criteria-06

[32] W. Zhang, W. Lei, and S. Liu, "Multipath Real-Time Transport Protocol Based on Application-Level Relay (MPRTP-AR)," Internet Engineering Task Force, Internet-Draft draft-leiwm-avtcore-mprtp-ar-07, Jan. 2017, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-leiwm-avtcore-mprtp-ar-07

[33] V. Singh, S. Ahsan, and J. Ott, "MPRTP: Multipath Considerations for Real-time Media," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 190–201. [Online]. Available: https://doi.org/10.1145/2483977.2484002

[34] G. Camarillo, "Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability," RFC 5694, Internet Engineering Task Force, Mar. 2009. [Online]. Available: https://tools.ietf.org/html/rfc5694

[35] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-peer Networks," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 189–202. [Online]. Available: https://doi.org/10.1145/1177080.1177105

[36] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144. [Online]. Available: https://doi.org/10.1145/1943552.1943572

[37] R. Castagno and D. Singer, "MIME Type Registrations for 3rd Generation Partnership Project (3GPP) Multimedia files," RFC 3839, IETF, Jul. 2004. [Online]. Available: https://tools.ietf.org/html/rfc3839

[38] H. Garudadri, "MIME Type Registrations for 3GPP2 Multimedia Files," RFC 4393, IETF, Mar. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4393

[39] G. Tian and Y. Liu, "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: ACM, 2012, pp. 109–120. [Online]. Available: https://doi.org/10.1145/2413176.2413190

[40] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: https://doi.org/10.1145/2155555.2155570

[41] R. Pantos, "HTTP Live Streaming," IETF, Sep. 2016. [Online]. Available: https://tools.ietf.org/html/draft-pantos-http-live-streaming-20

[42] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, Jun. 2011, pp. 1–8. [Online]. Available: https://doi.org/10.1109/WoWMoM.2011.5986186

[43] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, Apr. 2014. [Online]. Available: https://doi.org/10.1109/JSAC.2014.140403

[44] D. Yun and K. Chung, "Dynamic segment duration control for live streaming over HTTP," in *2016 International Conference on Information Networking (ICOIN)*, Jan 2016, pp. 206–210. [Online]. Available: https://doi.org/10.1109/ICOIN.2016.7427115

[45] T. Kupka, P. Halvorsen, and C. Griwodz, "An evaluation of live adaptive HTTP segment streaming request strategies," in *2011 IEEE 36th Conference on Local Computer Networks (LCN)*, Oct 2011, pp. 604–612. [Online]. Available: https://doi.org/10.1109/LCN.2011.6115524

[46] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015 - 2020 White Paper." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[47] K. J. Ma, R. Bartos, S. Bhatia, and R. Nair, "Mobile video delivery with HTTP," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 166–175, Apr. 2011. [Online]. Available: https://doi.org/10.1109/MCOM.2011.5741161

[48] A. Gouta, D. Hong, A. M. Kermarrec, and Y. Lelouedec, "HTTP Adaptive Streaming in Mobile Networks: Characteristics and Caching Opportunities," in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2013, pp. 90–100. [Online]. Available: https://doi.org/10.1109/MASCOTS.2013.17

[49] A. Gouta, C. Hong, D. Hong, A. M. Kermarrec, and Y. Lelouedec, "Large scale analysis of HTTP Adaptive Streaming in mobile networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, Jun. 2013, pp. 1–10. [Online]. Available: https://doi.org/10.1109/WoWMoM.2013.6583390

[50] K. J. Ma and R. Bartos, "HTTP Live Streaming Bandwidth Management Using Intelligent Segment Selection," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011, pp. 1–5. [Online]. Available: https://doi.org/10.1109/GLOCOM.2011.6133856

[51] C. Lai, R. Hwang, H. Chao, M. Hassan, and A. Alamri, "A buffer-aware HTTP live streaming approach for SDN-enabled 5G wireless networks," *IEEE Network*, vol. 29, no. 1, pp. 49–55, Jan 2015. [Online]. Available: https://doi.org/10.1109/MNET.2015.7018203

[52] C. Yang, Y. Li, and J. Chen, "A New Mobile Streaming System Base-On Http Live Streaming Protocol," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, Sept 2011, pp. 1–4. [Online]. Available: https://doi.org/10.1109/wicom.2011.6040621

[53] T. Wu, R. Huysegems, and T. Bostoen, "Scalable network-based video-freeze detection for HTTP adaptive streaming," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, Jun. 2015, pp. 95–104. [Online]. Available: https://doi.org/10.1109/IWQoS.2015.7404719

[54] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, "Offloading cellular networks with Information-Centric Networking: The case of video streaming," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, Jun. 2012, pp. 1–3. [Online]. Available: https://doi.org/10.1109/WoWMoM.2012.6263734

[55] R. Major and M. Hurst, "Apparatus, system, and method for adaptive-rate shifting of streaming content," Oct. 21 2014, uS Patent 8,868,772. [Online]. Available: https://www.google.com/patents/US8868772

[56] 3GPP, "Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)," Dec. 2012. [Online]. Available: https://www.qtc.jp/3GPP/Specs/26247-b10.pdf

[57] OIPF, "Open IPTV Forum (OIPF) Release 2 Specification Volume 2a - HTTP Adaptive Streaming," Jan. 2014. [Online]. Available: https://www.oipf.tv/docs/OIPF-T1-R2-Specification-Volume-2a-HTTP-Adaptive-Streaming-v2_3-2014-01-24.pdf

[58] DVB, "Digital Video Broadcasting (DVB);MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks," Jul. 2016. [Online]. Available: https://www.dvb.org/resources/public/standards/a168_dvb-dash.pdf

[59] DASH-IF, "DASH Industry Forum." [Online]. Available: http://dashif.org/

[60] ——, "Guidelines - Completed DASH-IF Interoperability Documents," 2016. [Online]. Available: http://dashif.org/guidelines/

[61] ——, "dash.js Player," 2016. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/wiki

[62] ——, "DASH-IF Test Assets Database," 2016. [Online]. Available: http://testassets.dashif.org/

[63] ——, "DASH Industry Forum - Clients," 2016. [Online]. Available: http://dashif.org/clients/

[64] ——, "DASH Industry Forum - Software," 2016. [Online]. Available: http://dashif.org/software/

[65] L. Popa, A. Ghodsi, and I. Stoica, "HTTP As the Narrow Waist of the Future Internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 6:1–6:6. [Online]. Available: https://doi.org/10.1145/1868447.1868453

[66] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing Cost and Performance for Content Multihoming," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. New York, NY, USA: ACM, 2012, pp. 371–382. [Online]. Available: https://doi.org/10.1145/2342356.2342432

[67] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A Case for a Coordinated Internet Video Control Plane," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. New York, NY, USA: ACM, 2012, pp. 359–370. [Online]. Available: https://doi.org/10.1145/2342356.2342431

[68] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 1, pp. 469–492, First quarter 2015. [Online]. Available: https://doi.org/10.1109/COMST.2014.2360940

[69] N. Cranley, P. Perry, and L. Murphy, "User Perception of Adapting Video Quality," *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, Aug. 2006. [Online]. Available: https://doi.org/10.1016/j.ijhcs.2005.12.002

[70] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang, "Inferring the QoE of HTTP Video Streaming from User-viewing Activities," in *Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack*, ser. W-MUST '11. New York, NY, USA: ACM, 2011, pp. 31–36. [Online]. Available: https://doi.org/10.1145/2018602.2018611

[71] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 362–373. [Online]. Available: https://doi.org/10.1145/2018436.2018478

[72] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A Quest for an Internet Video Quality-of-experience Metric," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 97–102. [Online]. Available: https://doi.org/10.1145/2390231.2390248

[73] ——, "Developing a Predictive Model of Quality of Experience for Internet Video," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 339–350. [Online]. Available: https://doi.org/10.1145/2486001.2486025

[74] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 4, no. 2, pp. 16:1–16:22, May 2008. [Online]. Available: https://doi.org/10.1145/1352012.1352020

[75] J. Yan, W. Muhlbauer, and B. Plattner, "Analytical Framework for Improving the Quality of Streaming Over TCP," *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1579–1590, Dec 2012. [Online]. Available: https://doi.org/10.1109/TMM.2012.2187182

[76] J. Yan, W. Mühlbauer, and B. Plattner, *An Analytical Model for Streaming over TCP*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 370–381. [Online]. Available: https://doi.org/10.1007/978-3-642-22875-9_34

[77] D. Gómez, F. Boronat, M. Montagud, and C. Luzón, "End-to-end DASH Platform Including a Network-based and Client-based Adaptive Quality Switching Module," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 38:1–38:4. [Online]. Available: https://doi.org/10.1145/2910017.2910638

[78] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 169–174. [Online]. Available: https://doi.org/10.1145/1943552.1943575

[79] ns2, "The Network Simulator - ns-2." [Online]. Available: https://www.isi.edu/nsnam/ns/

[80] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: ACM, 2012, pp. 97–108. [Online]. Available: https://doi.org/10.1145/2413176.2413189

[81] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[82] Adobe, "Open Source Media Framework (OSMF)." [Online]. Available: https://sourceforge.net/adobe/osmf/home/Home/

[83] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction," in *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference*, ser. SIGCOMM '16. New York, NY, USA: ACM, 2016, pp. 272–285. [Online]. Available: https://doi.org/10.1145/2934872.2934898

[84] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "CFA: A Practical Prediction System for Video QoE Optimization," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 137–150.

[85] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 325–338. [Online]. Available: https://doi.org/10.1145/2785956.2787486

[86] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 187–198. [Online]. Available: https://doi.org/10.1145/2619239.2626296

[87] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: https://doi.org/10.1145/2398776.2398800

[88] T.-Y. Huang, R. Johari, and N. McKeown, "Downtown Abbey Without the Hiccups: Buffer-based Rate Adaptation for HTTP Video Streaming," in *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, ser. FhMN '13. New York, NY, USA: ACM, 2013, pp. 9–14. [Online]. Available: https://doi.org/10.1145/2491172.2491179

[89] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *2012 19th International Packet Video Workshop (PV)*, May 2012, pp. 173–178. [Online]. Available: https://doi.org/10.1109/PV.2012.6229732

[90] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," *ArXiv e-prints*, Jan. 2016. [Online]. Available: https://arxiv.org/abs/1601.06748

[91] A. Beben, P. Wiśniewski, J. M. Batalla, and P. Krawiec, "ABMA+: Lightweight and Efficient Algorithm for HTTP Adaptive Streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 2:1–2:11. [Online]. Available: https://doi.org/10.1145/2910017.2910596

[92] VideoLAN, "VLC Media Player." [Online]. Available: https://www.videolan.org/vlc/

[93] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, "Buffer-based smooth rate adaptation for dynamic HTTP streaming," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, Oct 2013, pp. 1–9. [Online]. Available: https://doi.org/10.1109/APSIPA.2013.6694183

[94] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, Apr. 2014. [Online]. Available: https://doi.org/10.1109/JSAC.2014.140405

[95] C. Wang, A. Rizk, and M. Zink, "SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 1:1–1:12. [Online]. Available: https://doi.org/10.1145/2910017.2910593

[96] A. Mansy, B. Ver Steeg, and M. Ammar, "SABRE: A Client Based Technique for Mitigating the Buffer Bloat Effect of Adaptive Video Flows," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 214–225. [Online]. Available: https://doi.org/10.1145/2483977.2484004

[97] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, no. 11, pp. 40:40–40:54, Nov. 2011. [Online]. Available: https://doi.org/10.1145/2063166.2071893

[98] X. Yin, V. Sekar, and B. Sinopoli, "Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. New York, NY, USA: ACM, 2014, pp. 9:1–9:7. [Online]. Available: https://doi.org/10.1145/2670518.2673877

[99] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?" in *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video*, ser. NOSSDAV '12. New York, NY, USA: ACM, 2012, pp. 9–14. [Online]. Available: https://doi.org/10.1145/2229087.2229092

[100] L. D. Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *2013 20th International Packet Video Workshop*, Dec 2013, pp. 1–8. [Online]. Available: https://doi.org/10.1109/PV.2013.6691442

[101] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '13. New York, NY, USA: ACM, 2013, pp. 19–24. [Online]. Available: https://doi.org/10.1145/2460782.2460786

[102] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback Control for Adaptive Live Video Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 145–156. [Online]. Available: https://doi.org/10.1145/1943552.1943573

[103] C. Mueller, S. Lederer, and C. Timmerer, "A proxy effect analyis and fair adaptation algorithm for multiple competing Dynamic Adaptive Streaming over HTTP clients," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, Nov 2012, pp. 1–6. [Online]. Available: https://doi.org/10.1109/VCIP.2012.6410799

[104] S. Alcock and R. Nelson, "Application Flow Control in YouTube Video Streams," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 24–30, Apr. 2011. [Online]. Available: https://doi.org/10.1145/1971162.1971166

[105] K. Satoda, H. Yoshida, H. Ito, and K. Ozawa, "Adaptive video pacing method based on the prediction of stochastic TCP throughput," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 1944–1950.

[106] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, IETF, Sep. 2009. [Online]. Available: https://tools.ietf.org/html/rfc5681

[107] G. Fairhurst, A. Sathiaseelan, and R. Secchi, "Updating TCP to Support Rate-Limited Traffic," RFC 7661 (Experimental), Internet Engineering Task Force, Oct. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7661

[108] S. Nazir, Z. Hossain, R. Secchi, M. Broadbent, A. Petlund, and G. Fairhurst, "Performance Evaluation of Congestion Window Validation for DASH Transport," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA: ACM, 2014, pp. 67:67–67:72. [Online]. Available: https://doi.org/10.1145/2578260.2578275

[109] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis, "Trickle: Rate Limiting YouTube Video Streaming," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342821.2342838

[110] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824, Internet Engineering Task Force, Jan. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6824

[111] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 181–194. [Online]. Available: https://doi.org/10.1145/2663716.2663727

[112] B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure, "Are TCP Extensions Middlebox-proof?" in *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '13. New York, NY, USA: ACM, 2013, pp. 37–42. [Online]. Available: https://doi.org/10.1145/2535828.2535830

[113] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is It Still Possible to Extend TCP?" in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 181–194. [Online]. Available: https://doi.org/10.1145/2068816.2068834

[114] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon, "Cross-layer Scheduler for Video Streaming over MPTCP," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 7:1–7:12. [Online]. Available: https://doi.org/10.1145/2910017.2910594

[115] G. Fairhurst, B. Trammell, and M. KÃijhlewind, "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms," RFC 8095, Mar. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8095

[116] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 129–143. [Online]. Available: https://doi.org/10.1145/2999572.2999606

[117] Y. C. Chen, Y. S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A Measurement-based Study of MultiPath TCP Performance over Wireless Networks," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 455–468. [Online]. Available: https://doi.org/10.1145/2504730.2504751

[118] D. Jurca and P. Frossard, "Video Packet Selection and Scheduling for Multipath Streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 629–641, Apr. 2007. [Online]. Available: https://doi.org/10.1109/TMM.2006.888017

[119] Y.-C. Chen, D. Towsley, and R. Khalili, "MSPlayer: Multi-Source and multi-Path LeverAged YoutubER," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. New York, NY, USA: ACM, 2014, pp. 263–270. [Online]. Available: https://doi.org/10.1145/2674005.2675007

[120] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2345–2361, Sept 2016. [Online]. Available: https://doi.org/10.1109/TMC.2015.2497238

[121] S. McQuistin, C. Perkins, and M. Fayed, "TCP Goes to Hollywood," in *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '16. New York, NY, USA: ACM, 2016, pp. 5:1–5:6. [Online]. Available: https://doi.org/10.1145/2910642.2910648

[122] ——, "TCP Hollywood: An unordered, time-lined, TCP for networked multimedia applications," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, May 2016, pp. 422–430. [Online]. Available: https://doi.org/10.1109/IFIPNetworking.2016.7497221

[123] R. Houdaille and S. Gouache, "Shaping HTTP Adaptive Streams for a Better User Experience," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 1–9. [Online]. Available: https://doi.org/10.1145/2155555.2155557

[124] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH System," in *Proceedings of*

the 3rd Multimedia Systems Conference, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 11–22. [Online]. Available: https://doi.org/10.1145/2155555.2155558

[125] R. Rejaie and J. Kangasharju, "Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '01. New York, NY, USA: ACM, 2001, pp. 3–10. [Online]. Available: https://doi.org/10.1145/378344.378345

[126] W. Pu, Z. Zou, and C. W. Chen, "Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP," in *2012 19th International Packet Video Workshop (PV)*, May 2012, pp. 65–70. [Online]. Available: https://doi.org/10.1109/PV.2012.6229745

[127] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing Client Behavior of Commercial Mobile Video Streaming Services," in *Proceedings of Workshop on Mobile Video Delivery*, ser. MoViD'14. New York, NY, USA: ACM, 2013, pp. 8:1–8:6. [Online]. Available: https://doi.org/10.1145/2579465.2579469

[128] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, "Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-friendly Way," in *Proceedings of the 5th Workshop on Mobile Video*, ser. MoVid '13. New York, NY, USA: ACM, 2013, pp. 13–18. [Online]. Available: https://doi.org/10.1145/2457413.2457417

[129] D. Havey, R. Chertov, and K. Almeroth, "Receiver Driven Rate Adaptation for Wireless Multimedia Applications," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 155–166. [Online]. Available: https://doi.org/10.1145/2155555.2155582

[130] J. Kua, G. Armitage, and P. Branch, "The impact of active queue management on dash-based content delivery," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, Nov 2016, pp. 121–128. [Online]. Available: https://doi.org/10.1109/LCN.2016.24

[131] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," RFC 8033, Internet Engineering Task Force, Feb. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8033

[132] D. Taht, T. Hoeiland-Joergensen, P. McKenney, J. Gettys, and E. Dumazet, "The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm," IETF, Internet Draft draft-ietf-aqm-fq-codel-06, Mar. 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06

[133] R. Al-Saadi and G. Armitage, "Dummynet AQM v0.2 – CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 160418A, 18 April 2016. [Online]. Available: https://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf

[134] G. Hong, J. Martin, and J. M. Westall, "On fairness and application performance of active queue management in broadband cable networks," *Computer Networks*, vol. 91, pp. 390 – 406, 2015. [Online]. Available: https://doi.org/10.1016/j.comnet.2015.08.018

[135] A. Begen, K. Streeter, I. Bouazizi, and F. Denoual, "MPEG DASH Requirements for a webpush Protocol," Internet Engineering Task Force, Oct. 2014. [Online]. Available: https://tools.ietf.org/html/draft-begen-webpush-dash-reqs-00

[136] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426 (Informational), Internet Engineering Task Force, Jan. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7426

[137] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering Stable High-quality Video: An SDN Architecture with DASH Assisting Network Elements," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 4:1–4:10. [Online]. Available: https://doi.acm.org/10.1145/2910017.2910599

[138] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 3:1–3:12. [Online]. Available: https://doi.org/10.1145/2910017.2910597

[139] w3schools, "Browser Statistics," 2016. [Online]. Available: https://www.w3schools.com/browsers/

[140] Chromium, "SPDY: An experimental protocol for a faster web," 2010. [Online]. Available: https://www.chromium.org/spdy

[141] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540 (Standards Track), Internet Engineering Task Force, May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7540

[142] Chromium, "QUIC, a multiplexed stream transport over UDP." [Online]. Available: https://www.chromium.org/quic

[143] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," Internet Draft (Informational), Jul. 2016. [Online]. Available: https://tools.ietf.org/html/draft-hamilton-early-deployment-quic-00

[144] J. Iyengar and I. Swett, "QUIC Loss Recovery And Congestion Control," Internet Draft (Informational), Dec. 2015. [Online]. Available: https://tools.ietf.org/html/draft-tsvwg-quic-loss-recovery-01

[145] S. Wei and V. Swaminathan, "Low Latency Live Video Streaming over HTTP 2.0," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA: ACM, 2014, pp. 37:37–37:42. [Online]. Available: https://doi.org/10.1145/2578260.2578277

[146] C. Mueller, S. Lederer, C. Timmerer, and H. Hellwagner, "Dynamic Adaptive Streaming over HTTP/2.0," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2013, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICME.2013.6607498

[147] C. Müller and C. Timmerer, "A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP," in *Proceedings of the 19th ACM International Conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 723–726. [Online]. Available: https://doi.org/10.1145/2072298.2072429

[148] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: An Experimental Investigation of QUIC," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 609–614. [Online]. Available: https://doi.org/10.1145/2695664.2695706

[149] Christian Timmerer, "Advanced Transport Options for DASH: QUIC and HTTP/2," Jun. 2015. [Online]. Available: https://bitmovin.com/advanced-transport-options-dash-quic-http2/

[150] ——, "Advanced Transport Options for DASH: QUIC and HTTP/2 (Part II)," May 2016. [Online]. Available: https://bitmovin.com/advanced-transport-options-dash-quic-http2-part-ii/

[151] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," Dec. 2016. [Online]. Available: https://queue.acm.org/detail.cfm?id=3022184

[152] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 25:1–25:12. [Online]. Available: https://doi.org/10.1145/2079296.2079321

[153] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View from the Edge," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 15–28. [Online]. Available: https://doi.org/10.1145/1298306.1298310

[154] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, Mar. 2009. [Online]. Available: https://doi.org/10.1016/j.comnet.2008.09.022

[155] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 1–14. [Online]. Available: https://doi.org/10.1145/1298306.1298309

[156] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 345–360. [Online]. Available: https://doi.org/10.1145/2068816.2068849

[157] N. Carlsson and D. L. Eager, "Ephemeral content popularity at the edge and implications for on-demand caching," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, Oct. 2016. [Online]. Available: https://doi.org/10.1109/TPDS.2016.2614805

[158] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *INFOCOM, 2012 Proceedings IEEE*, Mar. 2012, pp. 1620–1628. [Online]. Available: https://doi.org/10.1109/INFCOM.2012.6195531

[159] V. K. Adhikari, S. Jain, Y. Chen, and Z. L. Zhang, "Vivisecting YouTube: An active measurement study," in *INFOCOM, 2012 Proceedings IEEE*, Mar. 2012, pp. 2521–2525. [Online]. Available: https://doi.org/10.1109/INFCOM.2012.6195644

[160] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting Video Server Selection Strategies in the YouTube CDN," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, Jun. 2011, pp. 248–257. [Online]. Available: https://doi.org/10.1109/ICDCS.2011.43

[161] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z. L. Zhang, "A tale of three CDNs: An active measurement study of Hulu and its CDNs," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, Mar. 2012, pp. 7–12. [Online]. Available: https://doi.org/10.1109/INFCOMW.2012.6193524

[162] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z. L. Zhang, M. Varvello, and M. Steiner, "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs," *IEEE/ACM Transactions on Networking*, vol. 23, no. 6, pp. 1984–1997, Dec 2015. [Online]. Available: https://doi.org/10.1109/TNET.2014.2354262

[163] D. K. Krishnappa, S. Khemmarat, L. Gao, and M. Zink, "On the Feasibility of Prefetching and Caching for Online TV Services: A Measurement Study on Hulu," in *Proceedings of the 12th International Conference on Passive and Active Measurement*, ser. PAM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 72–80. [Online]. Available: https://doi.org/10.1007/978-3-642-19260-9_8

[164] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 431–443. [Online]. Available: https://doi.org/10.1145/1879141.1879197

[165] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A Comparative Study of Android and iOS for Accessing Internet Streaming Services," in *Proceedings of the 14th International Conference on Passive and Active Measurement*, ser. PAM'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 104–114. [Online]. Available: https://doi.org/10.1007/978-3-642-36516-4_11

[166] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 157–168. [Online]. Available: https://doi.org/10.1145/1943552.1943574

Jonathan Kua (jtkua@swin.edu.au) received his Bachelor of Engineering (Telecommunication and Network Engineering) degree with First Class Honours from Swinburne University of Technology, Melbourne, Australia in 2014. He is currently a PhD candidate at the Internet For Things (I4T) Research Lab within the School of Software and Electrical Engineering at Swinburne University of Technology. His research interests include IP-based content delivery, adaptive multimedia streaming, data transport protocols and active queue management.

Grenville Armitage (garmitage@swin.edu.au) earned a B.Eng. in electrical engineering (Hons) in 1988 and a Ph.D. in electronic engineering in 1994, both from the University of Melbourne. He is a full professor of telecommunications engineering, was founding director of the Centre for Advanced Internet Architectures, and is founding Head of the Internet For Things (I4T) Research Lab at Swinburne University of Technology. He authored "Quality of Service In IP Networks: Foundations for a Multi-Service Internet" (Macmillan, April 2000) and co-authored Networking and Online Games "Understanding and Engineering Multiplayer Internet Games" (Wiley, April 2006). He is a member of IEEE, ACM and ACM SIGCOMM.

Philip Branch (pbranch@swin.edu.au) has a BSc and MTech from the University of Tasmania and a PhD in Computer Systems Engineering from Monash University. He is currently an Associate Professor within the School of Software and Electrical Engineering at Swinburne University. His research interests are in network design and analysis, and the applications and design of sensor networks. He is a co-author of "Networking and Online Games - Understanding and Engineering Multiplayer Internet Games" (Wiley, April 2006). He is a member of the IEEE and ACM.