JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Role of Dynamic Adaptive Streaming Over HTTP in Live Streaming

Samarth K

Shreedatta Nasik
Rakshitha Rajasekhara
Under the guidance of :
Dr. Jayasheela.C.S
Dept of IS&E, Banglore Institute of Technology

Abstract:

In this paper, we give an overview about the new DASH specification with a special focus on Live streaming services. In order to re-use existing web content distribution schemes, the new streaming technique provides the live stream as a sequence of files, which are continuously downloaded by the streaming client and also it offers adaptive and dynamic multimedia streaming solutions to heterogeneous end systems. However, it still faces many challenges in determining an appropriate rate adaptation technique to provide the best quality of experience (QoE) to the end systems. 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. These works are categorized according to the feedback signals used and the end-node that performs or assists with the adaptation.

Introduction:

The continuous and rapid increase of digital multimedia streaming is estimated to comprise 82% of Internet traffic by 2021. The proliferation of multimedia streaming devices (e.g., smartphones, laptops, tablets, gaming consoles, etc.) with wide accessibility to wireless networks plays a pivotal role in characterizing the increased demand for multimedia streaming over the Internet. The Hypertext Transfer Protocol (HTTP) is used in multimedia streaming without underlying network support via the Internet Protocol (IP). HTTP multimedia streaming provides device fluidity, cache-friendliness, data-agnostic web servers and the ability to avoid network translation and firewall problems. However, HTTP is based on the Transmission Control Protocol (TCP).

Therefore, having an overhead of about twice the media bit rate compared to the Real-Time Transport Protocol (RTP) and the User Datagram ProtocoDP). This leads to delays in delivering multimedia content and the under-

utilization of communication network resources. 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 behaviors, effective multimedia delivery presents many challenges.

Managing multimedia streaming for heterogeneous devices under unreliable network environments is challenging. Initially, a progressive download technique allowed HTTP applications to download a complete monolithic multimedia file to the end host using TCP. This results in all clients receiving the same video despite variations in network bandwidth and their capabilities. 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 Organization for Standardization (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 packetized by MPEG technologies.

Recently, MPEG-DASH (Dynamic Adaptive Streaming over HTTP) has become a standard that aims to provide an uninterrupted video streaming service to users with dynamic network ions 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 maximizing 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, Adobe HTTP Dynamic Streaming and Apple's HTTP Live Streaming. 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 categorize these rate adaptation techniques in terms of the feedback signals they use and the end-node that performs or assists the adaptation.

Background:

i. 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.

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 re-buffering 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 :

- Using a playout buffer: Short-term variations in network throughput can be overcome by using a playout buffer. 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.
- Transcoding-based solutions: This technique is the simplest to implement and used in CDNs. Raw video data is preprocessed to produce multiple encoded streams, each at a different bitrate, resulting in multiple versions of the same content. However, more storage and finer granularity of encoded bitrates are required to enable the client to optimize 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) and TCP Friendly Rate Control (TFRC) 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.

ii. 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) and Integrated Services (IntServ). 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. 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 as a protocol on top of UDP for realtime 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.

Multicast-based adaptive bitrate techniques 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. 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: P2P networks allow users to share content without the need for centralized servers, making it an attractive solution for delivering video over the Internet.

2. Peer-to-Peer (P2P) Streaming: There are two categories of P2P systems: tree-based and mesh-based. In tree-based systems, peers are organized 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". In mesh-based systems, peers connect to a random set of neighboring peers who watch the same content. Peers usually exchange information about their data availability and then they retrieve data from neighbors when it is ready. 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 DASH3 in the mid-2000s. 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 multisecond video chunks instead of a continuous stream of video packets.

Architectural overview:

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. In DASH systems (summarized 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, mime type 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). Video chunks are 3GP-formatted and in each Representation, there is a single initialization segment which contains the configuration data and many media segments. Concatenating the initialization 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. Unlike traditional streaming strategies, DASH does not control the video transmission rate directly.

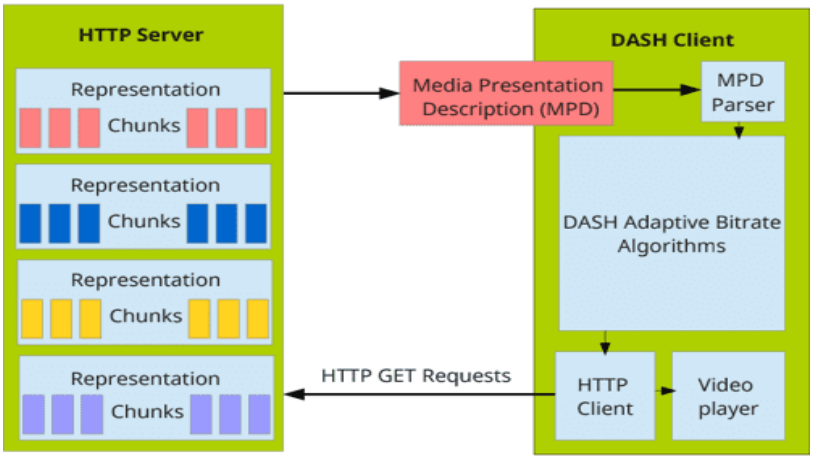

Figure 1

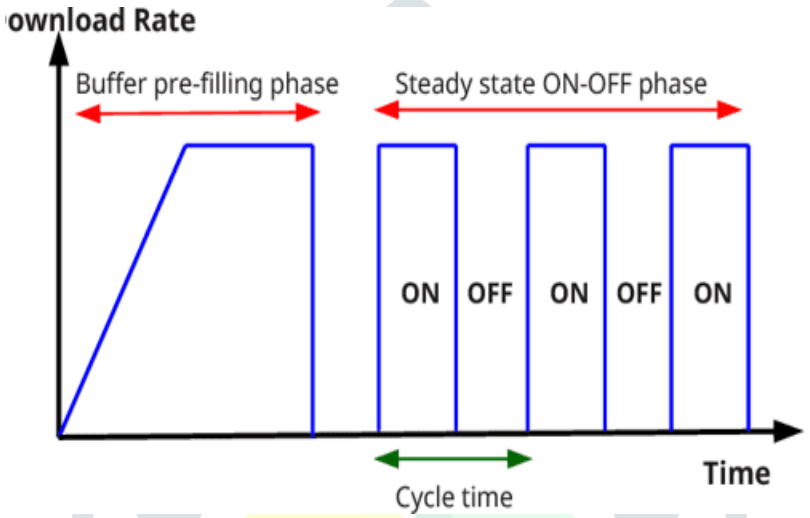

Figure 2

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).

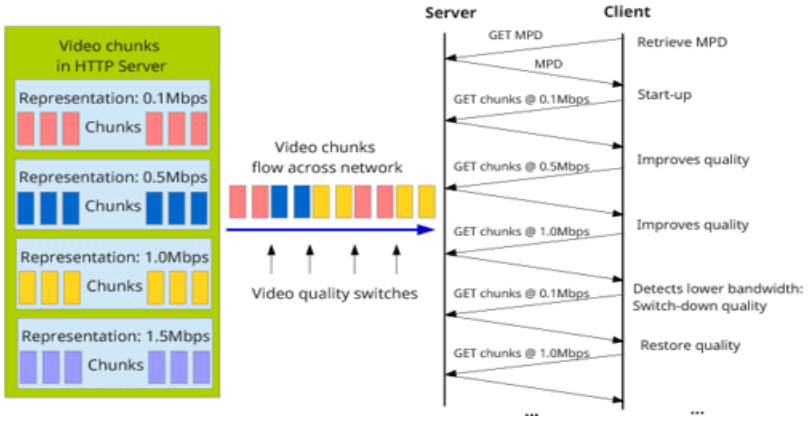

Figure 3

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. The video player uses various feedback signals observed for each chunk (such as recently achieved throughput4 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 . 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.

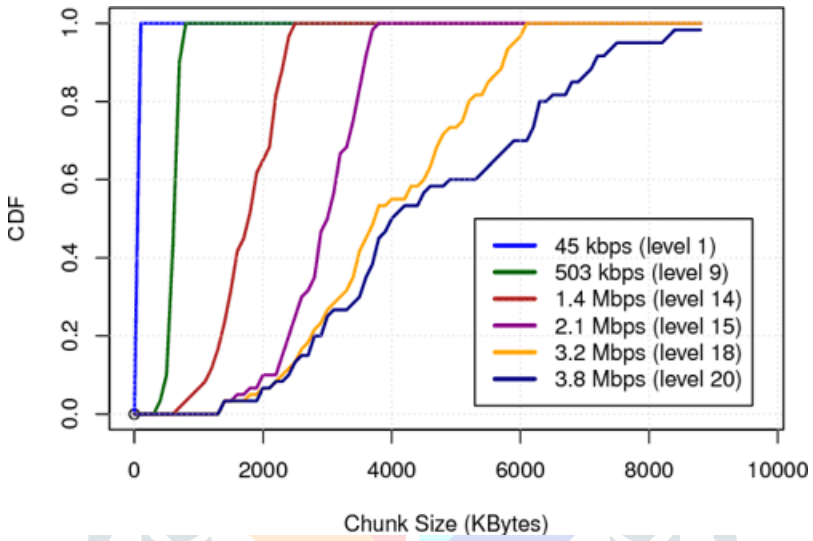

Figure 4

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.

When the client detects a reduction in bandwidth capacity (by utilizing 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 (an open-source, 9mins 46secs long animated video by the Blender Institute) 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.

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, despite the tighter latency constraints compared to on-demand streaming services. The challenge of live streaming is to minimize the time (end-toend 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 minimized 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. This technique is also used for inserting advertisements and commercials during live sessions. Client-side ABR algorithms used in on-demand streaming can be generally applied to live streaming. There are methods in controlling the chunk duration and HTTP request strategies in order to maintain a high liveliness and seamless playback of live content.

"Live streaming" entails the live broadcast of video content to viewers over the Internet. This practice has become dominated in most countries by the website Twitch. tv, which began broadcasting only video game content but has expanded to include a range of content, in the process attaining a market value of approximately $1 billion. Those who are able to bring in the largest crowds now make a living on the platform through a range of monetization techniques. Although most Twitch streamers are amateur producers, a significant number earn their living through streaming, whereas others are involved in associated activities such as commentating on "Esports" (competitive gaming) competitions.

Live streaming and game commentary are important for critical media studies for two reasons. First, live streaming is a major global phenomenon and expanding in scope and reach. The scale of the audience is now impressive enough to rival many television channels and traditional sports broadcasts, whereas it is also becoming an increasingly central element of digital gaming culture more broadly. Second, streaming represents a career path that many young people—disproportionately impacted by the financial crisis—are pursuing, finding the apparent opportunity to play games for a living understandably compelling (Johnson and Woodcock 2019). In a different vein, this is comparable with the opportunities of professional sports. These factors make Twitch an important element of contemporary youth employment dynamics, especially for those disaffected by, or unsuccessful in, traditional education or career paths. Our study of Twitch streamers highlights an emerging area within digital games that can be best understood through considering the labor that goes into the activities involved.

Our research combines interview and ethnographic data. We draw on hundred semistructured interviews conducted in 2016 and 2017 with professional and aspiring streamers and Esports commentators, of which approximately one third identified as women and two thirds as men. These interviews lasted between ten minutes and over one hour, and took place at gaming events in Poland, Germany, the United States, and the United Kingdom. The interviews focused on the themes of labor, performance, and work. Transcripts were coded, and interviewees have been assigned pseudonyms. The authors also carried out ethnographic observation at these gaming events, which provided context for the interviews. This research was supplemented by observation of two hundred unique live streams on Twitch from a diversity of streamers. This multi-sited ethnographic research helped to ground our analysis in not just the comments of live streamers as interview respondents but also their lived experience at these important and highly visible events during which they must perform and be publicly engaging for long stretches of time. The combination of methods provides much needed context to Twitch streaming culture and enables us to access participants in an online phenomenon that are otherwise difficult to reach.

We have sought to elaborate on some of the particularities of the labor process involved when someone begins to stream on the Twitch platform, specifically the labor of being presentable, funny, engaging, and in-character. This is a central part of the complex relationships on Twitch between the streamer, the game, the platform, and the audience. In doing so, these forms of labor have been shown to be deeply creative and contingent on being socially active and emotionally responsive. What emerges on this platform is a performativity of microcelebrity that is being mediated in novel ways: streamers seek to develop a unique manner of interaction with the audience, drawing on their own gameplay skills and personality and, in some cases, on the theatricality of character acting. To viewers, streams might seem casual and not that far from the experience of watching friends play on a console in a living room, yet this belies the striking amount of affective labor that goes into a stream. We have presented several of the most important elements of streamers' labor, but each of these is ripe for a more complete examination in its own right. In particular, the new pressures of being a professional streamer demand urgent attention. Given the rapid and ongoing growth of Twitch and the diverse forms of affective labor examined here, it is clear that understanding these forms of work on Twitch is central to theorizing digital game labor.

VR Streaming through Deep Reinforcement Learning Approach

In the last few years, the rapid development of fast video processing and omnidirectional cameras has bred a new media form, known as the immersive (3600 or panoramic) virtual reality (VR) video. Using user equipment (UE), such as head-mounted displays (HMDs), smartphones, and personal computers, immersive VR video can provide a 3600 omnidirectional immersive experience of, e.g., concerts, exhibitions, sports, etc. . The realization of immersive VR video relies on extremely large amount of data processing and transferring. Current VR systems depend largely on wired transmission, which restricts its applications, while wireless VR can potentially unleash its potential to the maximum.

In order to deliver immersive VR video over wireless networks, three fundamental challenges need to be urgently addressed. The first challenge lies that it is hard for the current cellular networks to provide sufficient high wireless transmission rate and thus to support the extremely high data rate requirement of immersive VR video transmission. The second major challenge lies in the heavy energy consumption in HMDs. The portion of an immersive VR video that a user is watching needs to be projected to a 2D plane referred to as the viewport. This portion mapping is called viewport rendering, which requires mapping the spherical VR video signal to the viewport pixel-by-pixel on an HMD, where complex matrix computation is needed and a large amount of energy will be consumed from the HMD's battery. The third challenge lies in the strict latency requirements imposed on the total delay of immersive VR video decoding, wireless transmission, and viewport rendering. The video decoding and viewport rendering operations typically take about $6 - 100$ ms, while wireless transmission will take $100 - 200$ ms. The large end-to-end latency will degrade the QoE of interactive immersive VR video playback significantly.

To overcome the challenge in supporting very high data rates, terahertz (THz) communication (0.1-10 THz), has been proposed as a promising enabler of super-high data rate, ultra reliable, and low delay applications, such as immersive VR video. Meanwhile, as a powerful supplement and enhancement of cloud computing, multi-access edge computing (MEC) enables HMDs to offload their energy demanding viewport rendering tasks to MEC servers (MECSs), and consequently offers an opportunity to tackle the last two challenges. However, since the problem of task offloading decision optimization is usually coupled with resource management, making the problem usually non-convex and NP-hard.

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. 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)

Gouta et al analyzed the behavior of mobile clients when streaming on-demand and live video from an ISP perspective. They also analyzed 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 Generalized Pareto Distribution best models the behavior.

Standardization:

Move Networks was an early proponent of HTTP-based adaptive streaming technology and was awarded a fundamental patent 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

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. Hence, commercial streaming services that use DASH implement their own proprietary techniques both for media representation and for client adaptation.

The DASH Industry Forum (DASH-IF) (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.

Benefits:

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.
- 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.
- 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

General ABR principles:

DASH uses ABR algorithms to select the appropriate video bitrates dynamically upon changing network conditions, which involves two control loops

- 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.

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 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).

Client-side Rate Adaptation Techniques:

This section presents a survey of the research literature on client-side rate adaptation techniques. Wang et al presented one of the most notable works in modeling multimedia streaming traffic over TCP in the early days of TCP-based streaming.

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 optimize 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.

Server-side, Transport layer and Network layer considerations:

There are many transport layer and network-level solutions for optimizing 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 optimizing server-side bitrate selection and congestion control. We exclude server side optimizations 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 serverside application layer and transport layer optimization to serve the pre-encoded video chunks stored in a regular Web server.

Conclusion:

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 clientend, most ABRs focused on client-end heuristics. However, there are also other techniques that utilize 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. 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 analyzing and characterizing 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 clientend 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, resource allocation and pacing TCP packets to smooth out traffic business.
- Fair resource sharing for DASH streams and other crosstraffic when multiple clients share a bottleneck. The content streaming community has some interesting challenges ahead.

References:

1. Multipath Dynamic Adaptive Streaming over HTTP Using Scalable Video Coding in Software Defined Networking by Ali Gohar and Sanghwan Lee.
2. Microsoft Silverlight Smooth Streaming, Microsoft, Redmond, WA, USA, 2016. Available: https://www.microsoft.com/silverlight/smoothstreaming/
3. C. Perkins and V. Singh, "Multimedia congestion control: Circuit breakers for unicast RTP sessions," Internet Eng. Task Force, Fremont, CA, USA, RFC 8083, Mar. 2017. Available: https://tools.ietf.org/html/rfc8083
4. W. Lei, W. Zhang, and S. Liu, "Multipath real-time transport protocol based on application-level relay (MPRTP-AR)," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-leiwm-avtcore-mprtp-ar-07, Jan. 2017. Available: https://tools.ietf.org/html/draft-leiwm-avtcore-mprtp-ar-07
5. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP by Jonathan Kua, Member, IEEE, Grenville Armitage, Member, IEEE, and Philip Branch, Member, IEEE
6. MEC-Assisted Immersive VR Video Streaming over Terahertz Wireless Networks from A Deep Reinforcement Learning Approach Jianbo Du, F. Richard Yu*, Fellow, IEEE, Guangyue Lu*, Junxuan Wang*, Jing Jiang, and Xiaoli Chu