ISSN 1512-1232

# A DASH Survey: the ON-OFF Traffic Problem and Contemporary Solutions

Koffka Khan[1], Wayne Goodridge[2]

[1] Department of Computing and Information Technology, The University of the West Indies, koffka.khan@gmail.com
[2] St. Augustine, Trinidad and Tobago, W.I.

***Abstract***

*This survey explores different aspects of adaptive streaming. It first gives a history of adaptive video streaming. It is desirable for users to receive desired or adequate quality of experience (QoE), when viewing an online video. ON-OFF traffic patterns associated with competing adaptive players is then described. This is a very important area of research as it is found that poor user-perceived QoE is experienced by competing players as a result of ON-OFF traffic patterns. A set of real-world factors which affect the user-perceived QoE as a result of competing players are then investigated. After stating the problem and factors that affect adaptive streaming players, this survey paper closes with a classification of state of the art adaptive video streaming techniques, which try to remedy the problem and some of the factors.*

***Keywords:*** *Adaptive, Streaming, Video, QoE, bottleneck, heuristic, stochastic, analytical.*

## 1. History of streaming methods

In the past ten years, the Internet has become a standard medium for multimedia delivery. The Internet today is prolific with applications that use video data. As a result, a number of video streaming services have been established over the past decade. Old-fashioned RTSP streaming eventually evolved to HTTP-based streaming protocols. This shift leads to a better user-perceived Quality of Experience (QoE). Progressive download uses HTTP as a protocol and succeeded traditional streaming. Today, HTTP adaptive video streaming has become the de facto standard. Real-time multimedia delivery has tight latency constraints, and data arriving too late is essentially useless. To create the illusion of motion, video frames should be played between 24-30 frames per second. 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. The variability in encoded bits per second leads to Variable Bit Rate (VBR) video. This is used by the ITU (International Telecommunication Union) H.264 and MPEG-4 video coding standards. The VBR encoded video is transmitted into the Internet. Since the Internet does not provide a constant, guar-anteed bandwidth for the video stream, the network can only support the video bitrate on a best-effort basis. QoS design is the fundamental functionality of the next generation IP router to enable differentiated delivery and to guarantee the delivery quality for diverse service traffic [1].

The importance of Quality of Service (QoS) is parallel with the recent evolution of telecommunication networks, which are characterized by a great heterogeneity [2]. All the applications that require a specific level of assurance from the network [2], especially Real -time video applications have quality-of-service (QoS) requirements [3].The Video streaming is often described as “bursty” and this can be attributed to the frame-based nature of video. Video frames are transmitted with a particular frame rate [4]. The analysis of a captured traffic from a "head and shoulders" multimedia telephony session shows us that voice and video packets have different characteristics. In fact, while voice packets have short and constant size, video packets have long and variable size [5]. If the network bandwidth is not sufficient to support the video bitrate, then the

ISSN 1512-1232
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, playout buffers and stream switching solutions has become industry standards. 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) [1] and TCP Friendly Rate Control (TFRC) [2] 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 in-crease/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.

### 1.1 Traditional streaming

Originally, video was streamed using stateful protocols. This became the traditional way of streaming. An example of a stateful protocol is Real-Time Streaming Protocol (RTSP). If a client wants to request data, it connects to the server. The server continuously monitors the state of the client after the connection is made. Communication is maintained using a continuous stream of data packets. These packets are sent to the client using either TCP or UDP. Applications run on top of transport protocols and real-time multimedia applications ideally trust the transport layer to minimize induced delays and deliver data with an appropriate degree of reliability and timeliness. Over the years, message-oriented transport proto-cols, such as Stream Control Transmission Protocol (SCTP) and Datagram Congestion Control Protocol (DCCP) [4] have been thought suitable. Nevertheless, the existence of Network Address Translations (NAT) [5], firewalls, and other middleboxes led to a range of known deployment challenges. However, when frames are partially or totally lost, UDP connections present less latency. Hence, most early work focused on enhancing User Datagram Protocol (UDP) [7] 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. The client to server connection stays open until the client disconnects. The player state between client and server, for example, stop, play or pause uses RTSP to communicate. In traditional streaming, data packets are streamed until the receive buffer is full. If the pause button is pressed, the client buffer is filled and the download is interrupted. The server does not resume the streaming of the video frames to the client anymore. This only happens when the play button is pressed again.

### 1.2 Progressive download

With stateful protocols memory is allocated dynamically at the server-side to keep information about all current connections. State information of connections that died out is cleaned up by the system. When there are a lot of client connections to the server some requests may not be processed as the server maximum number of clients is exceeded. Together with the increase in popularity of video streaming services and the performance issues with stateful protocols, the industry moved to stateless protocols to deliver video content to users. Each request is seen as an independent transaction in a stateless protocol. The client state is not constantly monitored. Hyper-text Transfer Protocol (HTTP) is the stateless protocol which is used in most of the modern video streaming applications. The HTTP [8] on top of Transmission Control Protocol (TCP) has become the primary

ISSN 1512-1232

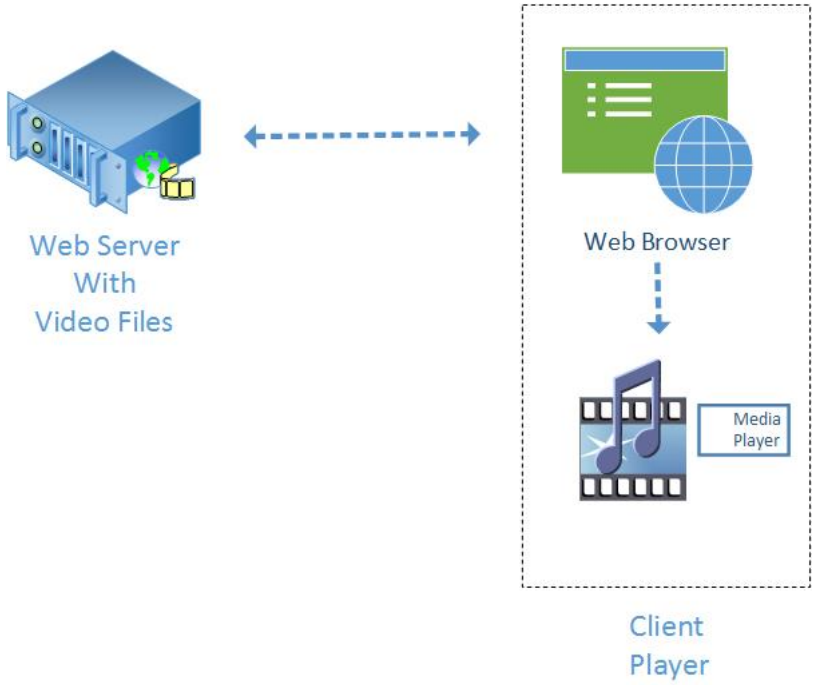


Figure 1: HTTP Download

protocol for multimedia content delivery over the Internet, also widely known as over-the-top (OTT) or Internet Protocol (IP)-based content delivery. HTTP avoids NAT and firewall traversal issues and provides reliability and deployment simplicity because of the widely implemented and deployed underlying TCP/IP protocol.

HTTP streaming, cf. Figure 1, uses a fast startup by downloading lowest quality and smallest segment first and adjusting its' rate afterwards. By using HTTP on top of TCP, Dynamic Adaptive Streaming over HTTP (DASH) yields the following benefits:

- Clients use the standard HTTP protocol which provides more ubiquitous reach as HTTP traffic can traverse NATs and firewalls [10].
- 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 [11].
- 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 [12], Adobe's HTTP Dynamic Streaming (HDS) [13] or Apple's HTTP Live Streaming (HLS) [14] leverage existing CDNs and proxy caches.

With progressive download the player takes over. The Web browser requests and receives a Meta File (a file describing the object) instead of receiving the entire video or audio file itself, cf. Figure 2. The Browser launches the appropriate player and passes it the Meta File. The Player sets up a connection with Web Server and downloads or streams the file. Client machines supporting HTTP can use progressive HTTP download. The original video file does not need to be downloaded completely. Hence, the client can start viewing the file immediately. This is similar to traditional streaming. However, there is a difference [15]. During a progressive download, the user can pause a video. When this happens, the download of the entire file still continues until the download is

ISSN 1512-1232
completed. With traditional streaming (RTSP), a pause command is sent to the server and the download is temporarily interrupted. Thus, progressive download wastes a lot of bandwidth if the user decides not to resume the video.

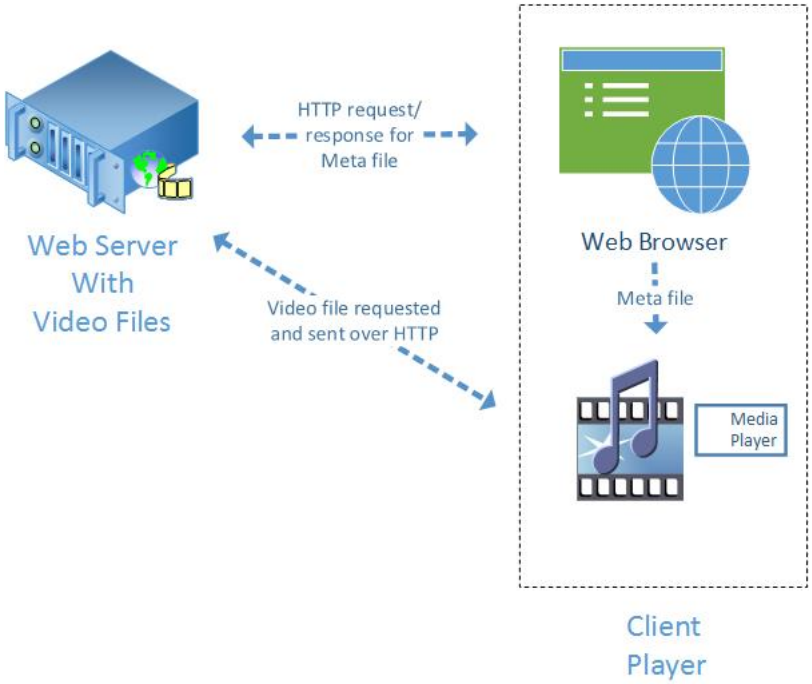


Figure 2: Progressive Download

### 1.3 Segmented streaming

The major disadvantage of progressive download is that a lot of unused data is sent over the network when the client decides to interrupt a streaming session. This is discussed in the previous section. The industry shifted to segmented streaming to avoid this waste of bandwidth, cf. Figure 3. The video file is split into smaller segments (chunks) of a certain length. Usually the segments range between 1 and 20 seconds. The segments are stored on the server. The sum of all the segments represents the total video for that specific range value, for example, a video may be broken up into 2 second and 4 second segments. Thus, there will be 2 second segments that when put together will form the entire video and so to for all the 4 second segments.

For each chunk, the client sends a separate HTTP request to the server. The server responds by sending the segment requested to the client. Each request and response for a particular segment is unique and standalone. Persistent connections are mostly used. Otherwise, a lot of overhead is introduced due to the TCP connection set-up. This is created when a separate connection has to be made for each segment request. To receive a next packet the client sends a new HTTP request. If the user decides to end or to pause a streaming session, the process is temporarily interrupted. In the majority of cases this occurs after the buffer is filled.

ISSN 1512-1232

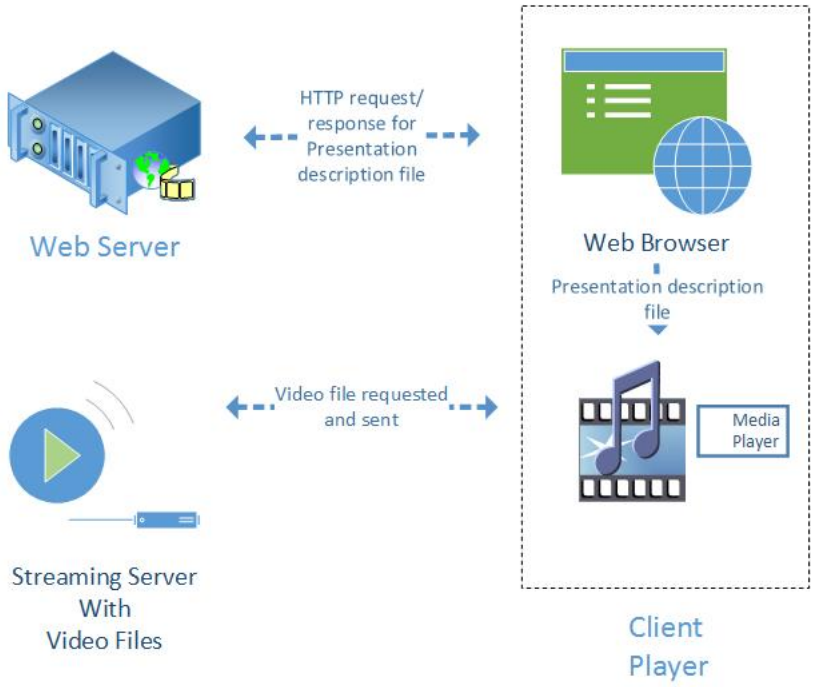


Figure 3: Segmented Streaming

The use of segmented streaming brings along some benefits. For example, when caching videos, segments at the beginning of a video are generally requested more than segments at the end. This happens as users do not always look at the entire video. The caching algorithm can decide to replace segments appearing at the end of the video first, by taking these differences in popularity into account.

### 1.4 HTTP Adaptive Streaming (HAS)

In progressive download and segmented streaming, the client receives a video file in a certain fixed quality. Different predefined qualities, for example, different bitrates, resolutions, quantization, etc...) are present and ready for download. These qualities are stored on the server. A decision is made about which quality to request, before the transmission starts. This takes into account the available bandwidth and the type of device used to display the video. Once the transmission is started, this quality is kept the same during the entire streaming session. There are no quality variations in the video sent to the client.

Variability of the network conditions during a streaming session implies that quality adaptation is needed. For example, consider a situation where the client streams a high quality video and where the quality cannot be adapted during the session. This can be the result of different network effects, for example, TCP long-lived flows [16]. When the bandwidth of the connection to the server drops, the segments will not be fully downloaded. Hence, the buffer runs out of video play-time. This leads to bothersome freezes in the video playback, which leads to stalling of the video.

A solution to the issue stated above is HTTP Adaptive Streaming (HAS), cf. Figure 4. With HAS, the client algorithm dynamically adapts the quality level. This adjustment is based on the current

ISSN 1512-1232

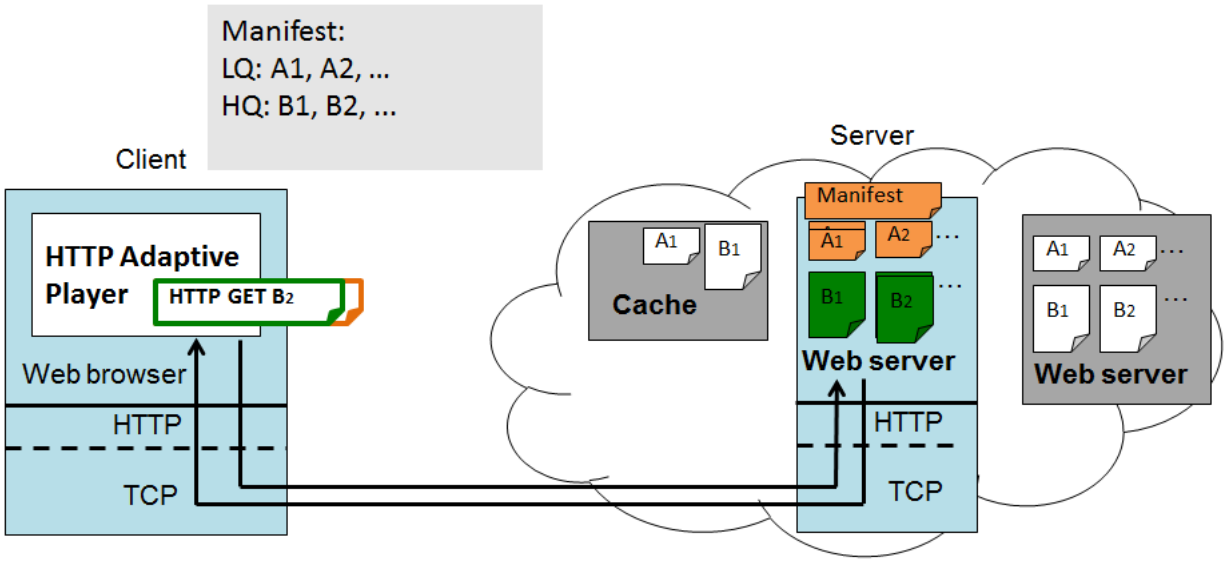


Figure 4: Adaptive Video Streaming

network conditions. This weaknesses of RTSP, progressive download and regular segmented streaming is ad-dressed by HAS. The media content needs to be preprocessed to allow this. The media content is split in several small segments of a certain length, as in segmented streaming. Each segment is now encoded in a number of predefined qualities. The different versions of the segments are then distributed over one or more media servers (HTTP web servers). It is important to note that each representation of the segments can be individually decoded. Also, the segment length determines the shortest video duration after which a quality adjustment can occur. Usually, this is 2 seconds in length.

A major decision-making challenge for the streaming client is which quality level to download from the server. This decision is required for each segment requested by the client during HAS streaming. However, all operations are transparent to the user as all bitrate selection happens behind the scenes. The viewer may notice a slight change in quality as the bitrates change, but no action is required on his/her part. Adaptive video streaming employs similar operating standards. However, there are some key differences. For instance, adaptive video streaming observes elements like (1) video buffer status to assess actual throughput, and (2) CPU utilization and dropped frames to evaluate the obtainable computing power on the playback station. This data is used to determine when to switch bitrates. For example, if the video buffer is full and CPU utilization low, the adaptive video streaming controller switches to a greater quality stream to augment the viewing experience. If the buffer drops under certain levels, or CPU use spikes above certain thresholds, the controller switches to a lower quality stream. Thus, adaptive video streaming enable producers to deliver outstanding quality bitrates at the high end of the bandwidth/power spectrum.

MPEG developed a HAS standard in 2012, Dynamic Adaptive Streaming over HTTP (DASH) [17], [18]. It is also known as MPEG-DASH, cf. Figure 4. It aims to provide an uninterrupted video streaming service to users with dynamic network conditions and heterogeneous devices. MPEG-DASH uses an application layer Adaptive Bitrate (ABR) 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. 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.

ISSN 1512-1232

- DASH requests and receives video data in terms of milli-second video chunks instead of a continuous stream of video packets.

Preprocessing is required by MPEG-DASH. In this process, a media presentation description (MPD) is created and hosted on a media server. In DASH-based systems video content is encoded into multiple versions. The differences are at discrete bitrates. Each encoded video is then fragmented into small video segments or chunks. Each segment contains a few seconds of video. Client can smoothly switch bitrates at the segment boundary. This is made possible by having segments from one bitrate aligned in a time-centric manner to the video segments from other bitrates. The Media Presentation Description (MPD) files describe content information. This information includes video profiles, metadata, codecs, byte-ranges, mimeType [19] server IP, addresses, and download URLs. URLs pointing to the video segments in an MPD can either be explicitly described or be constructed via a template [20]. Video chunks are 3GP-formatted [21]. Standard HTTP servers serve video segments and MPDs to clients.

DASH does not control the video transmission rate directly as with traditional streaming strategies. DASH clients depend on the underlying TCP algorithm to regulate the video transmission rate. This is determined by the congestion feedback from the client-server network path. At the start of a streaming session, the client requests the MPD file from the HTTP server. It then starts requesting video chunks in a sequential order. This process takes place as fast as possible to fill the playout buffer. Once this buffer is full, the player enters a steady state phase. During this phase, the player 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 when it waiting for another chunk download. This results in an alternating ON-OFF traffic pattern. The time between the start of two consecutive ON periods is termed a cycle time interval. The client typically keeps a few segments in the buffer to maintain smooth playback.

The video player uses various feedback signals observed for each segment. These system parameters, such as, average bandwidth and/or playout buffer underruns are used to determine which video rate to select for the next chunk to be downloaded. Consider average bandwidth as a system parameter. If the bandwidth is high, ABR should select a higher video rate to provide better QoE for the user. However, if the bandwidth is low, ABR should dynamically switch to a lower video rate to avoid playout buffer underruns. A good ABR algorithm is responsive to fluctuating network conditions and adapts smoothly to provide better QoE [22]. DASH clients cannot estimate the network bandwidth perfectly. The clients can only achieve the (discrete) video rates described by the MPD. It will select a rate below the estimated bandwidth to sustain video playback. When the network bandwidth exceeds the maximum video bitrate, the video rate is set to the maximum video bitrate. The smoothness between video bitrate transitions depend on the encoding granularity, that is, the number of video representations of the video content provided at the server.

An MPD file contains information on the available quality levels of the segments and other related meta-data information. This includes the URLs where each segment can be found as well as available bitrates, resolutions supported, bandwidth, frame rates, codecs, etc. A manifest file is the general term for an MPD in HAS.

The client has full control over the streaming session when using HAS. The rate adaptation algorithm is used to request the desired segment. It requests a specific quality by using a GET-operation. The adaptation algorithm takes care of adapting to network conditions and CPU capabilities. Usually the adaptation algorithm is in a module of the Controller within the adaptive player. Thus, some authors use the term adaptation algorithm and Controller to represent the part of the player that makes the request decision to the server. The requested chunks are played back smoothly. In current HAS implementations, MPEG-DASH and other (adaptive) video streaming services download segments of a single video from a single server. The client sends segment requests to this server even if there is a significant change in network conditions. This happens once

ISSN 1512-1232
the streaming session has started. On-demand and live streaming are the two types of streaming services that currently exist [23], [24], [25], [26]. On-demand streaming refers to requesting media content on the user's demand. The entire content is already present at one or more media servers and can be requested any time.

## 2. ON-OFF Traffic Patterns

Usually a streaming session is started by an adaptive video player and the player enters the Buffering-State [27]. The goal of the player in this state is to fill up its playback buffer as quickly as possible. Eventually the players' maximum buffer size is reached. To do this the player requests a new chunk in quick succession of the previously downloaded chunk. The player changes to Steady-State when the playback buffer size reaches a target value, for instance, 25 seconds. During streaming the players' goal is to maintain a constant playback buffer size. For the sake of simplicity, let us assume that a player can request one chunk every T seconds if the download duration is less than or equal to T. Otherwise, the player would request the next chunk as soon as all the data for the previous chunk is received. This can lead to an activity pattern in which the player is either in an ON period, downloading a chunk, or an OFF period, staying idle or waiting for the present chunk to be fully downloaded. The player estimates its values of the underlying network system parameters by using various estimation techniques. The discrete nature of the video bitrates makes it difficult for a client player to correctly perceive its fair-share bandwidth. This leads to video bitrate oscillation and other undesirable behaviours that negatively impact the user experience [28]. It may compute a running average of those parameters over time [28]. The player then uses these parameter values to select the bitrate for the next requested chunk. However, due to the time differences in various players' ON-OFF period the estimation of the system parameters may not be accurate. This result in five performance problems:

- buffer underruns
- network bandwidth utilization
- unfairness
- instability
- low quality

Based on these performance problems we define desired QoE as each competing player obtaining high buffer levels, bandwidth utilization, fairness, stability and quality.

Let us consider a simple model with three adaptive players sharing a bottleneck with bandwidth B. Assume that players are in Steady-State. There will be a request for a new chunk every T seconds. We ignore the TCP model and accept that a single connection gets the entire bandwidth B.

Let us assume that network bandwidth is shared equally. We let bi represents network bandwidth measured at the player. Thus, for equal sharing of the bandwidth the following condition holds: $b_1=b_2=b_3$. Therefore, $b_1+b_2+b_3=B$, anytime during the streaming process. Take the case where all players ON periods are non-overlapping during the download of a chunk, Figure 5. Each player would measure the bandwidth as B. Thus $b_1+b_2+b_3>B$. Hence, each player overestimates the bandwidth by a factor of three. The result is that players may request higher bitrates than the channel can provide. This can result in network congestion. Players will then measure that their bandwidth is less than their previous estimate. They will then switch back to a lower video bitrate. This creates a repeating oscillatory scenario, which causes instability.

ISSN 1512-1232

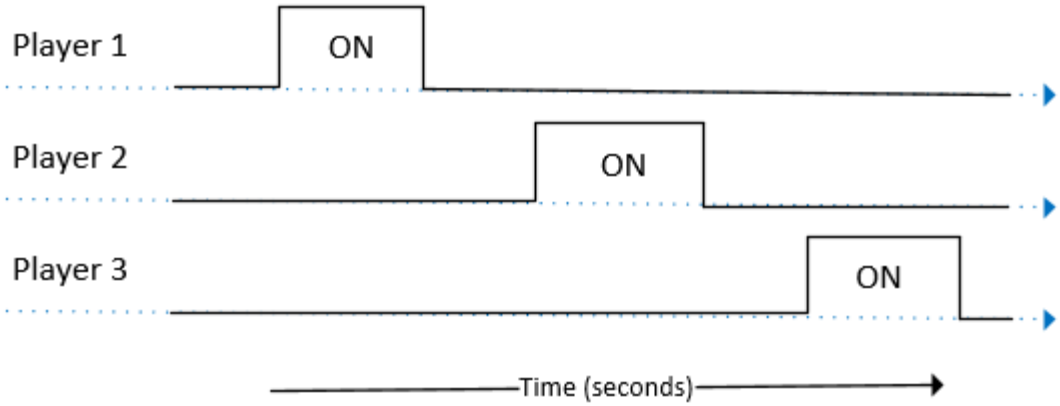


Figure 5: Non-overlapping ONs

The situation can arise where the ON period of one player falls within the ON period of the other player, Figure 6. This occurs if one player is requests a chunk with a low bitrate and another player requests a chunk with a high bitrate. The player requesting the lower bitrate estimates a bandwidth of B divided by 3, while the other player estimates a bandwidth that is more than B divided by 3. This means that player two overestimates the bandwidth. This overestimation by one of the three players can still result in the three players converging to a stable equilibrium. However, the player who estimates the higher bandwidth share will request a higher video bitrate. This will create an unfair bandwidth allocation to all players. The players who request low bitrates will experience buffer underruns and poor video quality due to flickering.

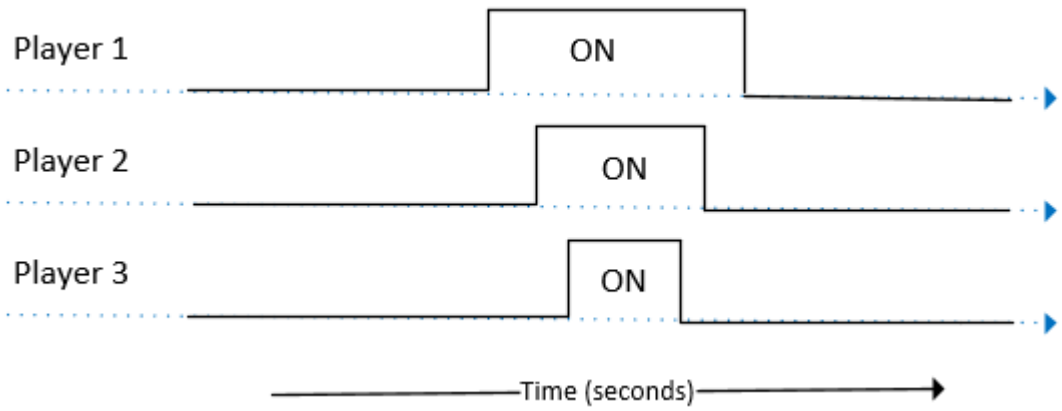


Figure 6: ON periods within each other

The situation can exist where the ON period of the three players are perfectly aligned, Figure 7. All players estimate bandwidth of B divided by 3. Thus, $b_1+b_2+b_3=B$. The three players estimate their bandwidth share correctly. However, bandwidth underutilization can still occur. To illustrate, suppose that the video has two quality levels, $q_1$ and $q_2$, respectively. Consider the cases among players where either the ON period of one player falls within the ON period of the other player or the ON period of the three players are perfectly aligned. Both cases can be stable if $b_1<B$ divided by 3, $b_2>B$ divided by 3, $b_3<B$ divided by 3, and $b_1+b_2+b_3<B$. However, the case where the ON period

ISSN 1512-1232

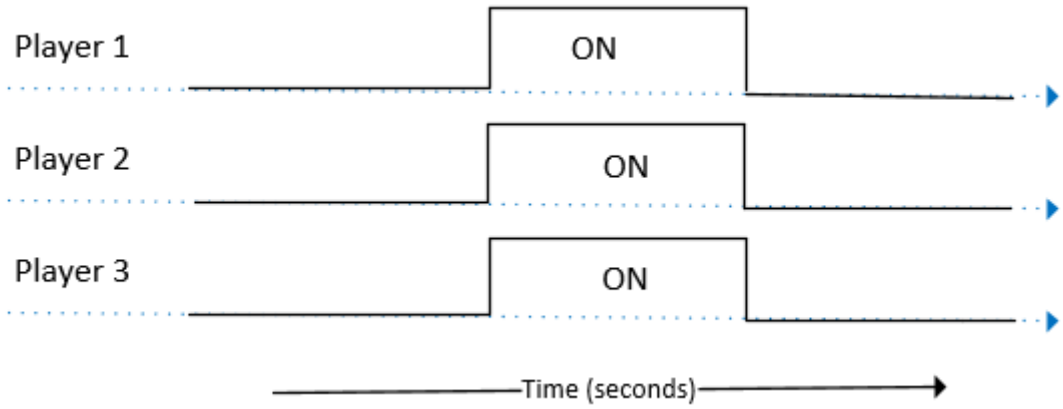


Figure 7: Perfectly Aligned ONs

of the three players are perfectly aligned, may result in all players request quality level, $q_1$. This causes bandwidth underutilization, even though it is stable and fair. The players who obtain low bandwidth will experience buffer underruns and poor video quality due to flickering.

## 3. Factors Affecting Adaptive Video Streaming

### 3.1 Time-varying bandwidth occurrences

Note that there is already competition for bandwidth at the bottleneck, which results in poor video quality shown to the user. Thus, when the bandwidth becomes too high or low in consecutive time periods within this competitive environment the quality can be severely degraded. As a result it is essential for our experiments to show the performance of players in network bottleneck competition with varying bandwidth.

### 3.2 Competition from other network flows, for example, TCP long-lived flows in networks

In [29] it has been established that the adaptive video players of three popular video streaming services were not able to get a fair share when coexisting with a TCP greedy flow. Authors name this issue the downward spiral effect and ascribe its cause to the on-off traffic pattern described above; authors suggest increasing the segment size and filter bandwidth estimates to tackle this issue. Chat and messaging (MSN, Skype) are served by TCP long-lived connections. Between computer-to-computer communications there are frequent “keepalive” message being transmitted periodically. This causes issues, such as, network resources being over-consumed. In addition, other issues occur during a TCP long-lived connection, including TCP congestion, and TCP connection recovery. The traffic features of TCP long-lived flows depends on specific applications usages characteristics.

In some cases, TCP is not able to fully utilize the transport or network layer resources because the application does not produce data fast enough. The application is producing small amounts of data at a relatively constant rate for the TCP layer. This results in small bursts of packets and in the extreme case a single packet of size less than the maximum segment size of the connection. Typical examples are live streaming applications such as Skype [30] that transfer data over TCP at a

ISSN 1512-1232

constant rate of 32 Kbit/s. Also, applications that use permanent TCP connections and send keep-alive packets during inactive periods, fall in this category (BitTorrent [31] exhibits this behavior during choke periods).

In some scenarios, the application is producing data in bursts separated from each other by idle periods. An example of such behavior is web browsing with persistent HTTP connections. The user clicks on a link to load a web page, causing a transfer period, reads the page, causing an idle period, and clicks on another link, causing another transfer period. It is observed that TCP long-lived flows may completely shut off TCP short-lived flows [32]. This causes performance problems for TCP short-lived flows, which generally carry interactive/delay sensitive data. Video data usually consists of TCP short-lived flows. It is because such flows are becoming increasingly dominant in Internet traffic patterns and the increase in competition with TCP long-lived flows makes it essential for a researcher to include as a part of their future research.

### 3.3 Players starting, pausing and stopping at different times

In most adaptive video streaming experiments it is assumed that the bottleneck bandwidth stays the same and the players have the same settings and initial states, which is a strong assumption. In reality players may have different initial state and can join at arbitrary time. Players can start following a certain statistic. It is therefore essential for researchers to evaluate their streaming algorithms in a settings where players are different (e.g., has different initial bitrate and buffer level).

### 3.4 Players downloading different videos

It is important for players to be tested in scenarios where videos with different sets of bitrate levels are downloaded from one or more servers. The adaptive approach should satisfy the various requirements of each video and user.

### 3.5 Users having different QoE functions

Scenarios where players have different QoE functions should be investigated by researchers. This is essential as different users may have different requirements when viewing a video. Usually researchers try to find the optimum QoE, which may be far from the actual requirements of some viewers. Subjective studies should be carried out to determine the best QoE for certain sections of users. Then the video QoE could be tailored for the specific QoE-based groups using adaptive streaming.

### 3.6 A large number of players competing for bandwidth

Scalability is a very important part of a video streaming algorithm. From a commercial point of view streaming is usually from a one-to-many model. Currently optimization approaches have difficulties in scaling and heuristics are used to counteract this problem.

## 4. State of the art Adaptive Video Streaming Techniques

Various techniques are used to provide solutions for adaptive video streaming. These adaptation techniques falls into three broad categories: (1) Heuristic, (2) Stochastic, and (3) Analytical. The

ISSN 1512-1232

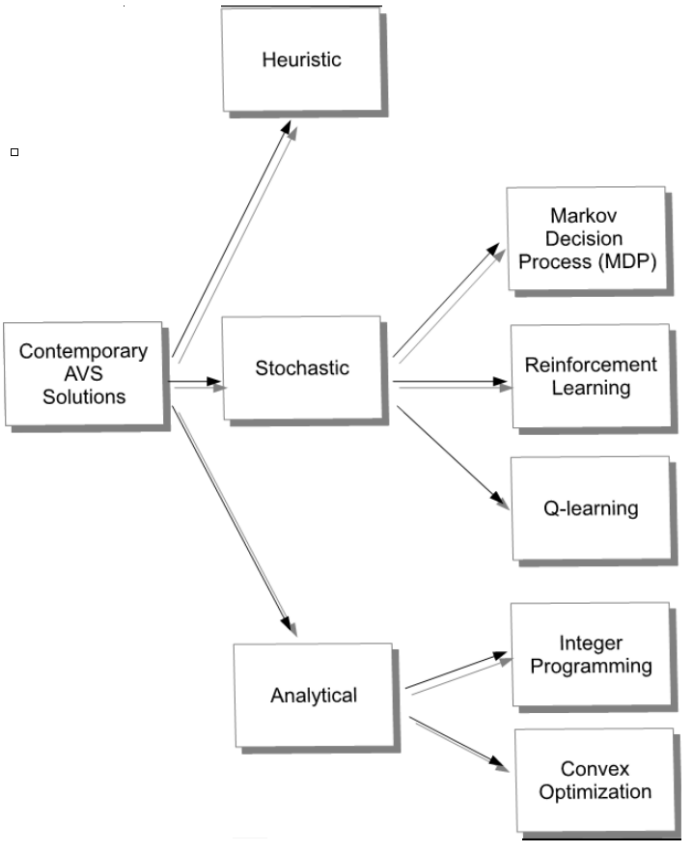


Figure 8: State of the art adaptive video streaming techniques

Heuristic techniques are described first and illustrate approaches that use a best guess technique to select segments. The probabilistic or stochastic techniques are then illustrated. These include Markov Decision Processes (MDPs), Reinforcement Learning (RL), and Q-learning (QL). Both RL and QL are described in terms of an MDP. Lastly analytical techniques are presented, which includes linear programming (LP) and convex optimization approaches.

### 4.1 Heuristic technique

Segment scheduling with stateless bitrate selection causes feedback loops, bad bandwidth estimation, bitrate switches and unfair bitrate choices. Researchers portray the FESTIVE control approach, and confirm that numerous problems occur when multiple bitrate-adaptive players share a bottleneck link. It uncovers the fact that the feedback signal the player receives is not a true reflection of the network state because of overlaying the adaptation logic over several layers. The following HTTP-based video delivery issues are investigated:

- the granularity of the control decisions,
- the timescales of adaptation,
- the nature of feedback from the network, and
- the interactions with other independent control loops in lower layers of the networking stack.

FESTIVE creates a robust video adaptation approach to achieve:

- Fairness - equal allocation of network resources,

ISSN 1512-1232

- Efficiency - highest bitrates for maximum user experience, and
- Stability - avoids needless bitrate switches.

The FESTIVE approach has the following features:

- Randomized segment scheduling: avoids sync biases in network state sampling,
- Stateful bitrate selection: compensates between biased bitrate and estimated bandwidth interaction,
- Delayed update: accounts for stability and efficiency tradeoff, and
- Bandwidth estimator: increases robustness to outliers.

The authors in [28] propose the PANDA approach. They observe that the discrete nature of the video bitrates makes it difficult for a client player to correctly perceive its fair-share bandwidth. This leads to video bitrate oscillation and other undesirable behaviours that negatively impact the user experience. The authors provide a solution to these problems at the application layer using a probe and adapt principle for video bitrate adaptation (where probe refers to trial increment of the data rate, instead of sending auxiliary piggybacking traffic). The authors outline a four-step state for a HAS rate adaptation approach:

- Estimating: guesses the network bandwidth that can legitimately be used,
- Smoothing: removes outliers,
- Quantizing: maps the continuous bit rate to the discrete video bitrate,
- Scheduling: selects the target interval until the next download request.

The advantages of PANDA are as follows. Firstly, as the bandwidth estimation by probing is quite accurate, one does not need to apply strong smoothing. Secondly, PANDA is very sensitive to bandwidth drops since after a bandwidth drop, the video bitrate reduction is made proportional to the TCP throughput.

ELASTIC [34] approach to adaptive client side streaming proposes a method that throttles the video level. This drives the playout buffer length to a set-point, which eliminates the ON-OFF traffic patterns. The player is always in ON phase. The basic concept is that two controllers are used. The first selects the video level to match the available bandwidth and the second controls the playout buffer length by ensuring that there is no gap between two consecutive segments. The ELAS-TIC approach produces a received video rate that oscillates around the fair share, with an increased number of video level switches. However, the main result shows that ELASTIC is able to get the fair share when competing with TCP long-lived flows.

Two content-aware adaptation schemes are proposed in [35]. They are Content-aware Probe and Adapt (CPANDA) and Content-aware Dynamic Programming (C-DP), which are able to decide the video bitrate for the next video segment, based on not only the available bandwidth, and the existing buffer capacity, but also the video content type. The content-aware adaptation schemes achieve better QoE when compared with the conventional PANDA scheme. The proposed schemes are able to optimize the QoE according to contents. They can achieve higher than acceptable QoE for most attractive contents such as those with high motion intensity to improve user experience for the most interesting scenes; select same video representations for segments belonging to a scene to avoid quality oscillation; and reserve buffer resource during low motion scenes

to avoid stalling for the following high motion scenes.

## 4.2 Stochastic technique

### 4.2.1 Markov Decision Processes

There is a growing body of literature on the use of Markov Decision Process (MDP) [36] to optimize video streaming. We now outline different ways MDPs are applied to adaptive video streaming. In [37] and [38], researchers found that bandwidth can vary severely in different locations. Adaptive streaming is modelled as an MDP problem to cope with varying network

ISSN 1512-1232
conditions [39]. The power consumption problem of video decoding can be effectively modelled as an MDP [40]. An MDP to optimize rate adaption of streaming video where the uncertainty in network bandwidth is modeled as a Markov chain with its own bandwidth states is given in [41] and [42]. In [43], and [44], a stochastic dynamic programming (SDP) [45] technique was proposed for rate adaption in DASH players, where the system rate is determined based on client buffer occupancy and bandwidth condition.

We label two studied models as MDP-DASH (Markov Decision Process DASH) and QC-DASH (Quality Control DASH). The goal of MDP-DASH [46] is to explore different methods to reduce decision making overhead for DASH-based adaptive video players. The states depend on the quality level of the downloaded chunk and the time available before the chunk playback deadline (current buffer occupancy measured in time). The actions (decisions) are the quality level of the next chunk to be downloaded. Higher rewards are given for watching a chunk in higher quality. There is a penalty for missing a deadline as well as switching quality from the present chunk to the next. For a given action (chunk size), state transition probabilities are calculated based on the Cumulative Distribution Function (CDF) [47] of the network bandwidth. The CDF allows calculation of the probability of a given buffer occupancy when the next chunk is downloaded. The buffer occupancy, together with the action (quality level decision), defines the next state. Transition probabilities will change with different CDFs. Different CDFs lead to different MDP strategies.

QC-DASH [43] uses Stochastic Dynamic Programming (SDP) to solve the MDP and aid adaptive video streaming. The three parameters to compute the state transition matrix are buffer level, average channel bandwidth and quality. The authors designed a cost function that penalizes situations that may lead to a reduction of the QoE. This computation is done offline, where the control policies map the environment information to the client requests. The main result is that the average quality requested with their algorithm is higher, but it also involves a related number of quality switches among segments.

### 4.2.2 Reinforcement Learning

In [48] a Reinforcement Learning-based [49] quality selection algorithm is proposed. When multiple players compete it is able to achieve fairness. A coordination proxy facilitates the coordination among players. Unlike current HAS heuristics, the algorithm is able to learn and adapt its policy depending on network conditions. No significant overhead is introduced into the network as fairness is achieved without explicit communication among agents. Researchers in [50] progressively maximize a pre-defined QoE-related reward function. By this action players are able to learn the optimal request strategy.

### 4.2.3 Q-Learning

A HAS client dynamically learns the optimal behavior corresponding to the current network environment [51] via Q-Learning [52]. A tunable reward function is used which considers multiple aspects of video quality. In order to optimize the QoE the HAS client dynamically learns the optimal behaviour corresponding to the current network environment [53]. The adaptive streaming problem can naturally be modeled as a Partial Observable Markov Decision Process (POMDP) [54] as the end-user has partial information about the network state based on the received throughput [55]. This service layer control mechanism gracefully degrades the video quality experienced by the end-user depending on the connection status. The end-user is able to find the most appropriate quality level for its stream.

ISSN 1512-1232

### 4.3 Analytical technique

#### 4.3.1 Integer Programming

A QoE-driven quality optimization approach is modeled as an Integer Linear Programming (ILP) [56] problem [57]. It maximizes the QoE over all clients. The ILP uses both centralized as well as distributed algorithms. The approach allows the client to take into account the in-network decisions during the rate adaptation process. It concurrently gives the player the ability to react to sudden bandwidth fluctuations in the local network. Hybrid Transmission strategies for DASH (HTD) in LTE network is considered in [58]. It considers both unicast and multicast modes for DASH. The optimization problem is formulated as a Mixed Binary Integer Programming (MBIP) problem. This together with a greedy algorithm improves the quality of experience (QoE) of wireless DASH users, and save the wireless resources in LTE networks.

#### 4.3.2 Convex optimization

Researchers in [59] propose a QoE evaluation model based on playback continuity, segment media quality and perceptual quality fluctuations caused by bitrate switching. They formulate the rate adaptation for DASH as a constrained convex optimization problem [60]. The design objective is to maximize the overall QoE for the end users while keeping the receiver buffer from under flowing. Near optimal tradeoff between overall QoE and playback continuity was observed.

An optimal content placement model maximizing the sum of user satisfaction of all contents is proposed in [61]. Due to the large number of content items in the network, the con-tent placement problem is usually a large-scale optimization problem. However, with mild assumption on the probability density function of the content popularity distribution it can be reduced. Here it is reduced to an equivalent small-scale convex problem. The number of variables is only the number of levels of representations.

## 5. Conclusion

A history of adaptive video streaming is given. This is followed by pointing out the key problem that occurs when adaptive players compete for bandwidth. This is the ON-OFF traffic patterns that results. Another area of interest is real-world factors that affect video streaming. These are pointed out and researchers can use these as test scenarios for experiments. Finally, a contemporary three-layer classification of streaming techniques is presented, which includes heuristic, stochastic and analytical techniques.

## References

[1] 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. IEEE, 1999, pp. 1337–1345.

[2] M. Handley, S. Floyd, J. Padhye, and J. Widmer, “Tcp friendly rate control (tfrc): Protocol specification,” Tech. Rep., 2002.

[3] L. Ong, “An introduction to the stream control transmission protocol (sctp),” 2002.

[4] E. Kohler, M. Handley, and S. Floyd, “Datagram congestion control protocol (dccp),” Tech. Rep., 2006.

[5] K. Egevang and P. Francis, “The ip network address translator (nat),” Tech. Rep., 1994.

ISSN 1512-1232
[6] A. Medina, M. Allman, and S. Floyd, “Measuring interactions between transport protocols and middleboxes,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, 2004, pp. 336– 341.
[7] J. Postel, “User datagram protocol,” Tech. Rep., 1980.
[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext transfer protocol–http/1.1,” Tech. Rep., 1999.
[9] J. Postel, “Transmission control protocol,” 1981.
[10] S. Guha and P. Francis, “Characterization and measurement of tcp traversal through nats and firewalls,” in Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement. USENIX Association, 2005, pp. 18–18.
[11] L. F. Sarmenta and S. Hirano, “Bayanihan: Building and studying web-based volunteer computing systems using java,” Future Generation Computer Systems, vol. 15, no. 5, pp. 675–686, 1999.
[12] A. Zambelli, “Iis smooth streaming technical overview,” Microsoft Corporation, vol. 3, p. 40, 2009.
[13] J. Ozer, Video Compression for Flash, Apple Devices and HTML5. Doceo Publishing, 2011.
[14] C. Timmerer and C. Muller,¨ “Http streaming of mpeg media,” Streaming Day, 2010.
[15] A. Begen, T. Akgul, and M. Baugher, “Watching video over the web: Part 1: Streaming protocols,” IEEE Internet Computing, vol. 15, no. 2, pp. 54–63, 2011.
[16] S. Ebrahimi-Taghizadeh, A. Helmy, and S. Gupta, “Tcp vs. tcp: a systematic study of adverse impact of short-lived tcp flows on long-lived tcp flows,” in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 2. IEEE, 2005, pp. 926–937.
[17] T. Stockhammer, “Dynamic adaptive streaming over http–: standards and design principles,” in Proceedings of the second annual ACM conference on Multimedia systems. ACM, 2011, pp. 133–144.
[18] S. Lederer, C. Muller,¨ and C. Timmerer, “Dynamic adaptive streaming over http dataset,” in Proceedings of the 3rd Multimedia Systems Conference. ACM, 2012, pp. 89–94.
[19] S. Pfeiffer, “Audio and video elements,” in The Definitive Guide to HTML5 Video. Springer, 2010, pp. 9–48.
[20] 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. IEEE, 2011, pp. 1–8.
[21] J. Kua, G. Armitage, and P. Branch, “A survey of rate adaptation techniques for dynamic adaptive streaming over http,” IEEE Commu-nications Surveys & Tutorials, 2017.
[22] S. Tavakoli, K. Brunnstrom,¨ K. Wang, B. Andren,´ M. Shahid, and N. Garcia, “Subjective quality assessment of an adaptive video streaming model,” in Image Quality and System Performance XI. SPIE, 2014.
[23] C. Huang, J. Li, and K. W. Ross, “Can internet video-on-demand be profitable?” ACM SIGCOMM Computer Communication Review, vol. 37, no. 4, pp. 133–144, 2007.
[24] M. Sjoblom,¨ M. Torh¨onen,¨ J. Hamari, and J. Macey, “Content structure is king: An empirical study on gratifications, game genres and content type on twitch,” Computers in Human Behavior, vol. 73, pp. 161–171, 2017.
[25] C. Zhang, J. Liu, M. Ma, L. Sun, and B. Li, “Seeker: Topic-aware viewing pattern prediction in crowdsourced interactive live streaming,” 2017.
[26] C. Koch, S. Hacker, and D. Hausheer, “Vodcast: Efficient sdn-based multicast for video on demand,” in A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on. IEEE, 2017, pp. 1–6.
[27] Y. Sani, A. Mauthe, and C. Edwards, “On the trajectory of video quality transition in http adaptive video streaming,” Multimedia Systems, pp. 1– 14, 2017.
[28] 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, 2014.

ISSN 1512-1232
[29] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over http," Signal Processing: Image Communication, vol. 27, no. 4, pp. 271–287, 2012.
[30] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," arXiv preprint cs/0412017, 2004.
[31] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in IPTPS, vol. 5. Springer, 2005, pp. 205–216.
[32] M. Mellia and H. Zhang, "Tcp model for short lived flows," IEEE communications letters, vol. 6, no. 2, pp. 85–87, 2002.
[33] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012, pp. 97–108.
[34] L. De 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. IEEE, 2013, pp. 1–8.
[35] S. Hu, L. Sun, C. Gui, E. Jammeh, and I.-H. Mkwawa, "Content-aware adaptation scheme for qoe optimized dash applications," in Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 1336–1341.
[36] E. A. Feinberg and A. Shwartz, Handbook of Markov decision processes: methods and applications. Springer Science & Business Media, 2012, vol. 40.
[37] P. Deshpande, X. Hou, and S. R. Das, "Performance comparison of 3g and metro-scale wifi for vehicular network access," in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010, pp. 301–307.
[38] J. Yao, S. S. Kanhere, and M. Hassan, "An empirical study of band-width predictability in mobile computing," in Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization. ACM, 2008, pp. 11–18.
[39] D. Jarnikov and T. Ozc¸elebi, "Client intelligence for adaptive streaming solutions," Signal Processing: Image Communication, vol. 26, no. 7, pp. 378–389, 2011.
[40] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. Van Der Schaar, "Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems," IEEE transactions on multimedia, vol. 15, no. 2, pp. 268–278, 2013.
[41] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in Proceedings of the 3rd multimedia systems conference. ACM, 2012, pp. 167–172.
[42] M. Xing, S. Xiang, and L. Cai, "Rate adaptation strategy for video streaming over multiple wireless access networks," in Global Commu-nications Conference (GLOBECOM), 2012 IEEE. IEEE, 2012, pp. 5745–5750.
[43] S. Garc´ıa, J. Cabrera, and N. Garc´ıa, "Quality-control algorithm for adaptive streaming services over wireless channels," IEEE Journal of Selected Topics in Signal Processing, vol. 9, no. 1, pp. 50–59, 2015.
[44] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over http with scal-able video coding," in Proceedings of the 3rd Multimedia Systems Conference. ACM, 2012, pp. 149–154.
[45] S. M. Ross, Introduction to stochastic dynamic programming. Academic press, 2014.
[46] A. Bokani, M. Hassan, and S. Kanhere, "Http-based adaptive streaming for mobile clients using markov decision process," in Packet Video Workshop (PV), 2013 20th International. IEEE, 2013, pp. 1–8.
[47] T. W. Anderson and D. A. Darling, "A test of goodness of fit," Journal of the American statistical association, vol. 49, no. 268, pp. 765–769, 1954.
[48] S. Petrangeli, M. Claeys, S. Latre,´ J. Famaey, and F. De Turck, "A multi-agent q-learning-based framework for achieving fairness in http adaptive streaming," in Network Operations and Management Sympo-sium (NOMS), 2014 IEEE. IEEE, 2014, pp. 1–9.

ISSN 1512-1232
[49] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press Cambridge, 1998, vol. 1, no. 1.
[50] T. Wu and W. Van Leekwijck, "Factor selection for reinforcement learning in http adaptive streaming," in International Conference on Multimedia Modeling. Springer, 2014, pp. 553–567.
[51] M. Claeys, S. Latre,´ J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a q-learning-based client quality selection algorithm for http adaptive video streaming," in Adaptive and Learning Agents Workshop, part of AAMAS2013 (ALA-2013), 2013, pp. 30–37.
[52] C. J. Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992.
[53] M. Claeys, S. Latre, J. Famaey, and F. De Turck, "Design and eval-uation of a self-learning http adaptive video streaming client," IEEE communications letters, vol. 18, no. 4, pp. 716–719, 2014.
[54] M. T. Spaan, "Partially observable markov decision processes," in Reinforcement Learning. Springer, 2012, pp. 387–414.
[56] J. Guo, X. Gong, J. Liang, S. Zhang, M. Zhao, W. Wang, and Z. Li, "A hybrid transmission approach for dash over mbms in lte network," in Global Communications Conference (GLOBECOM), 2015 IEEE. IEEE, 2015, pp. 1–6.
[55] D. Marinca, D. Barth, and D. De Vleeschauwer, "A q-learning solution for adaptive video streaming," in Selected Topics in Mobile and Wireless Networking (MoWNeT), 2013 International Conference on. IEEE, 2013, pp. 120–126.
[57] H. M. Wagner, "An integer linear-programming model for machine scheduling," Naval Research Logistics (NRL), vol. 6, no. 2, pp. 131– 140, 1959.
[58] N. Bouten, S. Latre, J. Famaey, W. Van Leekwijck, and F. De Turck, "In-network quality optimization for adaptive video streaming services," IEEE Transactions on Multimedia, vol. 16, no. 8, pp. 2281–2293, 2014.
[59] H. Zhang and X. Jiang, "A qoe-driven approach to rate adaptation for dynamic adaptive streaming over http," in Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on. IEEE, 2016, pp. 1–6.
[60] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2004.
[61] B. N. T. Duy, Q. A. Nguyen, P. L. Vo, and T.-A. Le, "Optimal content placement for adaptive bit-rate streaming in cache networks," in Information and Computer Science (NICS), 2015 2nd National Foundation for Science and Technology Development Conference on. IEEE, 2015, pp. 243-247.

Article received: 2017-07-23