CADLAD: Device-aware Bitrate Ladder Construction for HTTP Adaptive Streaming

Minh Nguyen[†], Babak Taraghi[†], Abdelhak Bentaleb[§], Roger Zimmermann[§], and Christian Timmerer[†]
[†]Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität Klagenfurt, Austria
[§]National University of Singapore, Singapore
{minh.nguyen,babak.taraghi,christian.timmerer}@aau.at, {bentaleb,rogerz}@comp.nus.edu.sg

***Abstract*—In this paper, we introduce a CMCD-Aware per-Device bitrate LADder construction (CADLAD) that leverages the Common Media Client Data (CMCD) standard to address the above issues. CADLAD comprises components at both client and server sides. The client calculates the top bitrate (`tb`) — a CMCD parameter to indicate the highest bitrate that can be rendered at the client — and sends it to the server together with its device type and screen resolution. The server decides on a suitable bitrate ladder, whose maximum bitrate and resolution are based on CMCD parameters, to the client device with the purpose of providing maximum QoE while minimizing delivered data. CADLAD has two versions to work in Video on Demand (VoD) and live streaming scenarios. Our CADLAD is client agnostic; hence, it can work with any players and ABR algorithms at the client. The experimental results show that CADLAD is able to increase the QoE by 2.6x while saving 71% of delivered data, compared to an existing bitrate ladder of an available video dataset. We implement our idea within CAdViSE — an open-source testbed for reproducibility.**

***Index Terms*—*HTTP Adaptive Streaming*, QoE, live streaming, CMCD, bitrate ladder construction**

I. Introduction

Video streaming has occupied a major part of Internet traffic. The first half of 2021 observed nearly 54% of overall traffic coming from video streaming applications [1]. According to Bitmovin Inc. report [2], a variety of devices are used to stream multimedia content. TV, desktop, and mobile are the three most used devices for watching video content [3], [4]. They have different sizes and resolutions that may affect the end user experience. End users would not recognize the quality difference between 1080p and 4K video if they are using mobiles [5].

HTTP Adaptive Streaming (HAS) has become a crucial technique for the delivery of multimedia content [6], [7]. In HAS, the video, stored at the server side, is encoded at multiple representations based on a bitrate ladder. A list of these representations is described in a Media Presentation Description (MPD) metadata file (*e.g.*, bitrate, resolution, codec). Each representation is then chopped into fixed-length segments with 2s to 10s duration. A HAS client first downloads the MPD file to extract the information about the video, especially the bitrate ladder. An Adaptive Bitrate (ABR) algorithm located at the client is in charge of selecting suitable bitrates from the bitrate ladder according to its resources (*e.g.*, buffer occupancy) and/or network conditions (*e.g.*, throughput).

Traditional HAS servers utilize static bitrate ladders with a fixed number of representations for both Video on Demand (VoD) and live streaming, neglecting the device types (*e.g.*, TV, desktop, and mobile) and end users' networks. One of the reasons is the lack of information transferred between the client and the server. This makes some representations rarely selected, especially when the network throughput (*e.g.*, from 3G networks) is lower than the bitrate of some available representations (*e.g.*, 4K resolution). This results in additional encoding costs for those bitrate levels and unnecessary storage costs. Additionally, some buffer-based ABR algorithms such as BBA-0 [8] often tend to select the highest bitrate in the bitrate ladder when the buffer exceeds a threshold. This strategy possibly leads to rebuffering events when the throughput suddenly drops while the selected bitrate does not ensure a recognizable improvement in quality, especially for small devices.

The Consumer Technology Association (CTA) released the Common Media Client Data (CMCD) specification in 2020 [9] to enable the client to provide the server with more information related to playback, *e.g.*, buffer length, and the highest bitrate rendered. Though CMCD offers a lot of client-side information to the server, there are two research questions that need to be answered:

- *How are the values of CMCD parameters determined?*
- *How are those values utilized by the server to improve HAS' performance?*

Answering these questions, in this paper, we take advantage of CMCD parameters to determine the suitable bitrate ladder for particular device types. Our proposed approach, CADLAD, calculates the top bitrate (`tb`) parameter (defined in the CMCD specification) based on the estimated throughput at the client-side and sends it to the server along with other information. At the server side, CADLAD gathers the `tb` values from all the clients in the streaming session and creates MPD files with suitable bitrate ladders for specific (group of) client(s), based on the device characteristics (*i.e.*, device type and screen resolution) of those clients. Multiple MPD files are generated in a streaming session and clients can only receive the ones that are suitable for their devices and network characteristics. We design two versions of CADLAD at the server side so that it can be used for VoD and live streaming scenarios. Additionally, CADLAD works with any HAS-based video

978-3-903176-51-5

player.

The contributions of our work are two-fold:

- We propose a CMCD-aware per-device bitrate ladder construction for the purpose of improving the QoE while saving delivery bandwidth cost.
- We publish an open-source reference implementation of CADLAD for reproducibility.

II. Background, Related Work and Motivation

A. Common Media Client Data

Common Media Client Data (CMCD) has been recently standardized by the Consumer Technology Association (CTA) [9]. CMCD provides a mechanism for data that is collected by the client to be delivered to the server as a custom HTTP request header, an HTTP query argument or JSON object. The description of CMCD parameters can be found in [9].

In this paper, we propose to add two additional parameters to the CMCD protocol: *(i)* device type (`dt`), and *(ii)* screen width (`sw`). Device type (`dt`) is a token that expresses the type of device used to watch the video. `dt` can take one of the three values: t, d, m that represent TV, desktop, and mobile, respectively. Screen width (`sw`) indicates the width of the screen's resolution in pixels. These two parameters will be used in our proposed approach to determine a suitable bitrate ladder for each client.

Bentaleb *et al.* [10] built the first proof-of-concept system that conforms to the CMCD specification at both the client and server sides. The authors deployed a buffer-aware bandwidth allocation approach to improve the QoE of clients sharing network bandwidth. Begen *et al.* [11] have provided an overview of the CMCD specification and described several applications of CMCD to enhance HAS. No implementations or evaluations are given in this work. Unlike these works, we propose a CMCD-aware per-device bitrate ladder design that leverages the `tb` parameter of CMCD to limit the highest bitrate in the MPD file with the objective of reducing downloaded data while improving the QoE for the end user.

B. Bitrate Ladder Construction

There have been several attempts to construct an optimal bitrate ladder for HAS.

The simplest way is to use a fixed bitrate ladder with a pre-defined encoding configuration. Multiple video streaming platforms have released their recommendations for bitrate ladders, including bitrate values and their corresponding resolutions [12]–[14]. There are two main limitations of this approach. First, the content dependency is not considered; thus, a specific bitrate can result in a high perceptual quality for easy-to-encode videos but low quality for others. Second, the device and network conditions of the end user are not taken into account. Encoding content that is mostly watched on small screen devices such as mobiles at high bitrate levels will lead to storage waste if those levels are never selected or if they experience multiple video stalls when the throughput is unfavorable.

TABLE I: Apple ABR ladder.

Resolution (p)	234	360	432	540	720	1080
Bitrate (Mbps)	0.14	0.36	0.73, 1.1	2	3, 4.5	6, 7.8

Algorithm 1: Bitrate Ladder Construction for VoD streaming.

Input: sw_n, tb_n
Output: $\mathcal{L}_n$
for *i = N, N-1, …, 1* **do**
  if $w_i \leq sw_n$ **then**
    $R_n^u = r_i$;
    break;
for $r = R_n^u, \ldots, r_1$ **do**
  if $r \leq tb_n \times (1 + \mu)$ **then**
    $R_n = r$;
    break;
$\mathcal{L}_n = \{r_1, \ldots, R_n\}$;
return $\mathcal{L}_n$;

Al-Issa *et al.* [15] provide max-min bounded bitrate guidance for HAS over Software-Defined Networking (SDN)-enabled networks. Their approach, named BBGDASH, comprises three layers: the application layer, the control layer, and the infrastructure layer to select the maximum and the minimum allowed bitrate levels for specific players. Then, the end user invokes her ABR scheme to select the final bitrate within the range of maximum and minimum bitrate levels. Despite its success, BBGDASH relies on features and capabilities of SDN that would limit its implementation in real-world environments.

Recently, per-title encoding schemes have been investigated to provide an optimized bitrate ladder for each video content. The work in [16], [17] determine the convex-hull to select the optimal bitrate at a specific resolution with the objective of maximizing the visual quality. Though these schemes can improve the visual quality, convex-hull determination requires many test encodings as each resolution is tested with multiple bitrate levels. Thus, they are suitable for VoD streaming only. Menon *et al.* [18] proposed a low-latency **O**nline **P**er**T**itle **E**ncoding (OPTE) to improve bitrate ladders for live video streaming. OPTE exploits spatial and temporal features of the video content to predict the most suitable resolution for every bitrate level. Despite their promising performance, per-title encoding schemes have not taken into account the device and network characteristics. This results in the encoding of unnecessary bitrate levels which are not requested by the end-user.

Meanwhile, our proposed approach, CADLAD, determines suitable bitrate ladders for the clients based on their device types and network conditions (*i.e.*, throughput) to improve the QoE while reducing the downloaded data.

III. CADLAD Approach

Our proposed CADLAD approach comprises two elements:

1) **CMCD Parameters Determination**. This element is located at the client side. Its output is the values for CMCD parameters and in our case, we calculate `tb` which represents the highest bitrate in the MPD file. We also add two additional parameters: `dt`, which shows the type of the device used by the client and `sw` for the width of screen resolution of that device (see Section II).
2) **Bitrate Ladder Construction**. This element is deployed at the server side to determine the suitable bitrate ladder for each client based on the client's device information (*i.e.*, device type and screen resolution) and `tb`.

The details of the two elements are given as follows.

A. CMCD Parameters Determination

The CMCD Parameters Determination element is implemented at the client side to select information about the device (*i.e.*, device type, and screen width), and to calculate the top bitrate `tb`.

CADLAD determines `tb` according to the recent throughput; hence, this value varies during the streaming session. Again, `tb` is used to inform the server which is the highest bitrate required by the client. Here, we propose a simple but efficient method to determine the value for this parameter. To avoid rebuffering events in case of small buffer size, the client should not download a bitrate that is much higher than the throughput. Therefore, CADLAD limits the maximum bitrate in the bitrate ladder to the average throughput. Then, the client can select a lower-quality representation in case of unfavorable throughput and avoid downloading unnecessarily high bitrates when the throughput suddenly goes up. Let t_j denote the measured throughput after downloading the j^{th} segment. CADLAD set tb as the average throughput of the last L segments as follows.

$$tb = \frac{1}{L} \times \sum_{i=0}^{L-1} t_{j-i} \tag{1}$$

The value of L should not be too large to adapt fast to the throughput fluctuation but also not be too small to avoid frequent changes in the bitrate ladder at the server. In this paper, we set $L = 5$ in our experiments with 4 s-segments. Other values of L with different segment lengths will be considered in future work.

B. Bitrate Ladder Construction

Assume that C clients are joining a streaming session. The n^{th} client pushes the following CMCD parameters to the server: dt_n ($\in \{t, d, m\}$), sw_n, and tb_n. In this paper, we design CADLAD for two scenarios of streaming: *(i)* VoD streaming, and *(ii)* live streaming.

1) VoD streaming: In this scenario, the content is already encoded into multiple representations at the server side. Adding all representations into the MPD file might make the client select an unnecessary high bitrate, especially for small-screen devices and volatile throughput. Therefore, it is useful to provide dynamic MPD files to the client to help it make a better decision on the requested bitrate. Here, we use the `tb` parameter of CMCD to periodically limit the highest bitrate in the MPD file based on the device types and the current throughput.

Let $\mathcal{L} = \{r_1, r_2, \ldots, r_N\}$ denote the set of all pre-defined bitrates r_i ($i \in [1, N]$) of the video at the server side. Each bitrate r_i has its own width of resolution w_i (in pixel). Upon receiving the CMCD parameters from client n, CADLAD in VoD mode determines the highest bitrate R_n in the MPD file for that client which satisfies the following conditions.

First, we define the upper bound R_n^u of the bitrate ladder for client n with device type dt_n and screen width sw_n as in Eq (2).

$$R_n^u = \max\{r_i | w_i \leq sw_n, r_i \in \mathcal{L}\}. \tag{2}$$

The width of resolution of upper bound R_n^u does not exceed the resolution of the device at the end user. The highest bitrate R_n should not be greater than R_n^u or in other words, it satisfies the following constraint

$$R_n \leq R_n^u. \tag{3}$$

Second, due to the network condition expressed by the value of tb_n, the highest bitrate in the bitrate ladder R_n should not exceed tb_n with a margin of μ, $\mu \in [0, 1]$ as shown in Eq (4).

$$R_n \leq tb_n \times (1 + \mu) \tag{4}$$

The margin μ is added here to allow the ABR at the end user device to improve the video quality when the current throughput becomes better. In this paper, we set $\mu = 0.1$.

Then the server picks all available bitrate levels that are less than or equal to R_n into the bitrate ladder $\mathcal{L}_n$ and prepares the corresponding MPD file before sending it to the client. Our algorithm for determining the bitrate ladder $\mathcal{L}_n$ for the client n is described in Algorithm 1.

2) Live streaming: In live streaming scenarios, the video content is encoded during the streaming session. According to the CMCD parameters (*i.e.*, device type, screen width, and top bitrate), CADLAD gathers the end user devices into groups, then determines the highest bitrate for the bitrate ladder of each group before encoding the video content.

A problem occurring here is that the top bitrate tb of the end user devices may be significantly different from each other due to the network connection types (*e.g.*, 3G, 4G, 5G, and WIFI) or the network conditions. Thus, selecting the highest bitrate for each end user results in a vast number of representations to encode. We solve this problem by putting multiple clients into groups based on their device types. Each group of clients can be further split into k clusters depending on the server capacity. More clusters will result in more bitrates to encode, which requires larger server capacity. Each cluster is represented by

a bitrate which will be the highest bitrate in the MPD file for the end users of that cluster.

We denote C^t, C^d, and C^m as the number of TVs, desktops, and mobiles joining the streaming session, respectively. Here, CADLAD utilizes a k-means clustering algorithm [19] with the CMCD parameters as the input data to select the k clusters in each group of devices. A cluster i is represented by a centroid c_i determining the top bitrate tb_{c_i} and the corresponding resolution width sw_{c_i} in the bitrate ladder.

After determining the top bitrate $tb^{dt}_{c_i}$ in the ladder for cluster i with device type dt, CADLAD picks representations based on a predefined bitrate ladder (*e.g.*, the Apple bitrate ladder [12] or the YouTube bitrate ladder [13]) whose bitrates and resolution widths are less than or equal to tb_{c_i} and sw_{c_i} of centroid c_i, respectively. This operation is similar to Algorithm 1.

The overview of CADLAD at the server side is shown in Fig. 1.

IV. Implementation and Evaluation

A. System Implementation

To evaluate our proposed improvements in a streaming session using CMCD we have extended CAdViSE [20] and used it as our testbed[1]. In order to enable the CMCD function, we have used the dash.js player (v4.3.0) and added the new parameters as stated in Section II. Modifying the JavaScript source code of the player allows the collection of the real-time values for those parameters and includes them in the header of each request being directed to the server. We did not introduce any decision-making logic to the clients and we used the default dash.js ABR algorithm, *Dynamic*. However, to gather the required data from the player we did multiple modifications on the client docker containers that were used in CAdViSE. At the servers, which are also being deployed into Amazon Web Services (AWS) using docker containers, we retrieved the data from requests headers and used k-means clustering to aggregate the data and inform the server of the current requirements in terms of encoded bitrate and resolution based on the characteristics of the current devices that are connected to that server.

The test sequence is "Seconds that count" with a 322-second length from [21] that supports up to 8K resolutions. However, as dash.js does not yet support 8K, we use up to 4K resolutions (*i.e.*, 2160p) for this paper. The full bitrate ladder is $\mathcal{L}$=$\{$100, 200, 375, 550, 750, 1000, 1500, 3000, 5800, 7500, 12000, 17000$\}$kbit/s with corresponding resolutions $\{$144, 180, 216, 288, 360, 432, 576, 720, 1080, 1440, 2160, 2160$\}$p.

We use two network traces: *(i)* 4G network and *(ii)* Cascade, each of which represents a specific scenario of a video streaming session. The Cascade network trace periodically changes the throughput after every 15 seconds among predefined values $\{200, 100, 50, 25, 50, 100, 200, 100, ...\}$(Mbps). The 4G network trace collected in [22] has the average of $11\,352\,\mathrm{kbit/s}$, standard deviation of $9404\,\mathrm{kbit/s}$ and maximum value of $48\,874\,\mathrm{kbit/s}$.

We take into account three device types (screen resolution) of the clients: (i) *TV (2160p)*, (ii) *desktop (1080p)*, (iii) *mobile (720p)*[2]. We assume all clients with the same device type have the same screen size. Different screen sizes of a specific device type will be investigated in our future work. Each experiment is repeated five times for the accuracy, and the experimental results in the next sections show the average values.

We consider the following evaluation metrics:

- Bitrate (BR): The average bitrate of all segments downloaded by same-device end users in a streaming session.
- Number of switches (#SW): The average number of switches of same-device end users in a streaming session.
- Stall duration (SD): The average period while the video is frozen at same-device end users.
- QoE score (QoE): The QoE score calculated by model ITU-T P.1203 mode 1[3] [23], [24].

We use the `HASClipStitcher`[4], a tool that utilizes CAdViSE log records of the streaming sessions to calculate these metrics.

B. Experimental Results

In this section, we evaluate CADLAD for both VoD and live streaming. We investigate different device types in our experiments. CADLAD-T, CADLAD-D, and CADLAD-M denote all clients in the experiments that are TVs, desktops, and mobiles, respectively, and CADLAD is enabled. CADLAD-A means all device types are joining a single streaming session. We use CADLAD-A in the live streaming scenario in which ten clients, including 4 TVs, 3 desktops, and 3 mobiles, participate in a live streaming session. We denote dashjs4 when all clients are not using CMCD and CADLAD. Dashjs4 clients can download a maximum of 4K resolution.

1) VoD Streaming: In the VoD streaming scenario, the videos on the server side have been already encoded into multiple bitrate levels. Thus, each client sends an MPD request and immediately receives an MPD file based on its CMCD parameters. In this case, we consider client and network metrics including #SW, SD, and BR. A single client is joining the streaming session with the 4G network trace [22]. Fig. 2 compares the performance of CADLAD, compared with the default dash.js v4.3.0 player (dashjs4) in VoD streaming.

It can be seen that CADLAD outperforms dashjs4. The dashjs4 player suffers from a long stall duration of 55s, whereas the TV clients with CADLAD (CADLAD-T) have, on average, 35.3s of stalls. Those dashjs clients with CADLAD-T can play up to 4K resolution (Fig. 2a). This is because CADLAD limits the maximum bitrate in the MPD file to the average values of recent throughput, especially when the throughput is unstable. Thus, CADLAD-T avoids downloading high bitrate levels that can lead to stall events when the

[1]https://github.com/cd-athena/CMCD-CAdViSE. Access: 05 July 2022.

[2]https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12. Access: 06 September 2022.

[3]https://github.com/itu-p1203/itu-p1203 Access: 05 July 2022.

[4]https://github.com/cd-athena/HASClipStitcher. Access: 05 July 2022.

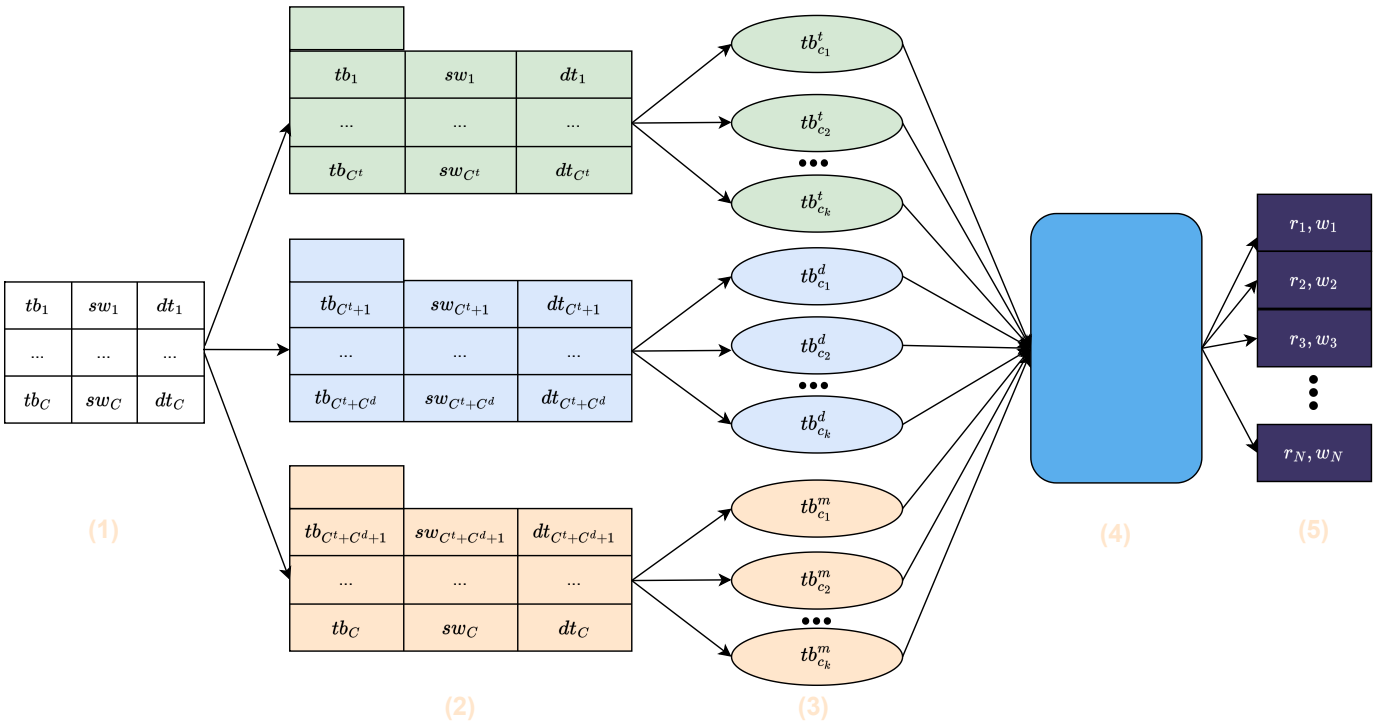
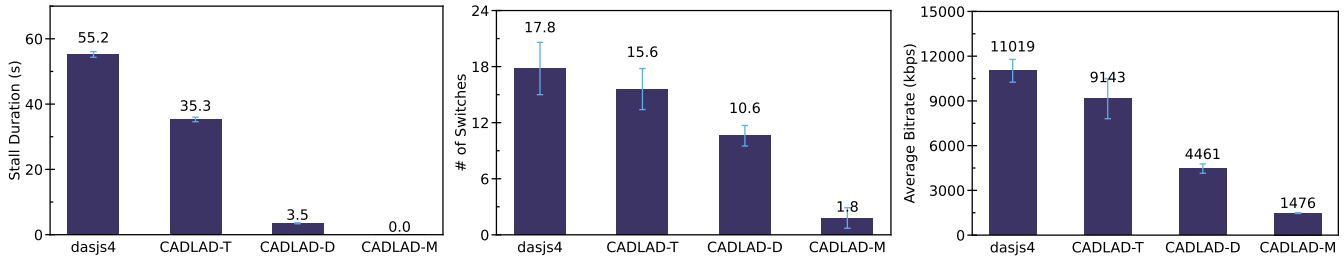


Fig. 1: Overview of CADLAD at the server side in live streaming. (1): The server collects all CMCD parameters from the clients for a specific streaming session. (2): CADLAD classifies sets of CMCD parameters into three groups: TV, Desktop, and Mobile. (3): In each group, k-means clustering is utilized to decide the maximum bitrates for k sets of clients. (4): Bitrate ladder selection component determines which bitrates and corresponding resolutions will be used in video encoding. (5): A set of bitrates and resolutions is chosen to encode the video and create the MPD file.



(a) Stall duration. (b) Number of switches. (c) Average bitrate.

Fig. 2: Experimental results for a single client in VoD with the 4G network trace.

throughput drops. Desktop and mobile clients with CADLAD (*i.e.*, CADLAD-D and CADLAD-M, respectively) can decrease the stall duration substantially with no more than 3.5s in each streaming session. This is attributed to the lower bitrate required for these smaller devices. Low bitrate levels help CADLAD-D and CADLAD-M ramp up the buffer occupancy, which allows the client to deal with throughput fluctuations.

In Fig. 2b, due to a lower stall duration, CADLAD in three types of devices, *i.e.*, CADLAD-T, CADLAD-D, and CADLAD-M, decreases the number of switches from 12% to 90%, compared with dashjs4 without CMCD. In addition, the smaller the devices, the fewer the number of switches. This value for CADLAD-T is 15.6 switches per streaming session, whereas CADLAD-D and CADLAD-M experience only 10.6 and 1.8 switches on average.

Fig. 2c and Table II show that our proposed approach saves a remarkable amount of downloaded data while providing substantial improvements in the QoE. A dashjs4 client without CMCD downloads an average bitrate of $11\,019\,\mathrm{kbit/s}$ for each segment and its QoE score is only 1.5 if a TV device is used. In contrast, clients using CADLAD can achieve better QoE while significantly reducing the downloaded data. For instance, CADLAD-T needs an average bitrate of $9143\,\mathrm{kbit/s}$ for each segment, which is 17% less than with dashjs4, and it attains a slightly higher QoE score of 1.6. Due to smooth playback without any stalls, CADLAD-M downloads an average bitrate

TABLE II: QoE score of the compared approaches in VOD streaming with different types of devices.

Device	Approach	QoE Score
TV	**dashjs4**	1.5
	CADLAD-T	1.6
Desktop	**dashjs4**	1.6
	CADLAD-D	3.9
Mobile	**dashjs4**	1.6
	CADLAD-M	4.3

TABLE III: QoE score of the compared approaches in live streaming with different types of devices.

Device	Approach	QoE Score
TV	**dashjs4**	1.6
	CADLAD-T	2.1
Desktop	**dashjs4**	1.7
	CADLAD-D	4.3
Mobile	**dashjs4**	1.76
	CADLAD-M	4.5
All types	**dashjs4**	1.66
	CADLAD-A	2.4

of $1476\,\mathrm{kbit/s}$ while providing the viewer a QoE score of 4.3. On the contrary, the QoE score of dashjs4 is only 1.6 on a mobile device.

From the above analysis, it can be seen that CADLAD significantly improves the performance of a player through a higher QoE score while saving data.

2) Live Streaming: We consider multiple clients joining a single live streaming session. Ten clients are used in each experiment. They can be *(i)* all dashjs4 clients without CMCD, *(ii)* all TV clients with CADLAD (*i.e.*, CADLAD-T), *(iii)* all desktop clients with CADLAD (*i.e.*, CADLAD-D), *(iv)* all mobile clients with CADLAD (*i.e.*, CADLAD-M), or *(v)* a mix of 4 clients of CADLAD-T, 3 clients of CADLAD-D, and 3 clients of CADLAD-M, collectively referred to as CADLAD-A. In each live streaming session, 10 clients are sharing the Cascade network trace, which is similar to the setup in [10]. A 4G network is not suitable for multiple clients as the trace was collected for single device usage only. In this scenario, as the video is encoded while being streamed, it is necessary for the server to reduce redundant bitrate levels that are not often requested from the clients. Here, k-means clustering is utilized in CADLAD to obtain this goal.

Fig 3 compares the results of CADLAD and dashjs4 with different clients. It can be seen that CADLAD outperforms dashjs4 in most evaluation metrics. Dashjs4 clients without CMCD experience an average of $26.6\,\mathrm{s}$ of stall events, whereas this number for CADLAD-T is 20% less with only $21.3\,\mathrm{s}$. All CADLAD-D and CADLAD-M clients can play the video smoothly without any stalls. As explained earlier, CADLAD limits the maximum bitrate for desktops (*i.e.*, $5800\,\mathrm{kbit/s}$ at 1080p) and mobiles (*i.e.*, $3000\,\mathrm{kbit/s}$ at 720p), which directs the clients to download suitable bitrates for their devices and avoid unnecessary high bitrate levels like $17\,000\,\mathrm{kbit/s}$ at 4K. This strategy allows the clients to cope with throughput fluctuations. In a scenario of all types of devices joining a live streaming session, CADLAD-A, which has 4 TVs, 3 desktops, and 3 mobiles, suffers only $14.7\,\mathrm{s}$ of stall duration, 45% shorter than dashjs4.

In terms of quality switches in Fig. 3b, CADLAD in general provides better results. CALDAD-D, CADLAD-M, and CADLAD-A have an average of 4.5, 0.9, and 6.6 switches on each client, respectively, whereas dashjs4 clients experience 14.5 quality switches. CADLAD-T has more switches than the others, with 16.6 quality changes. This can be explained by the update time in CADLAD. In this experiment, we set the `minimumUpdatePeriod` parameter to $20\,\mathrm{s}$ in the MPD file which means the client will request for a new MPD file after every $20\,\mathrm{s}$. As the throughput changes every $15\,\mathrm{s}$, CADLAD might update late. In other device types, CADLAD-D and CADLAD-M do not suffer from many switches as they require lower bitrate levels than CADLAD-T. Due to small screen resolutions, CADLAD-D and CADLAD-M download a maximum of $5800\,\mathrm{kbit/s}$ at 1080p and $3000\,\mathrm{kbit/s}$ at 720p, respectively, which can be delivered smoothly to the client by the Cascade network in the experiments. That is also the reason for significantly fewer switches of CADLAD-A with a mix of device types, compared to CADLAD-T and dashjs4.

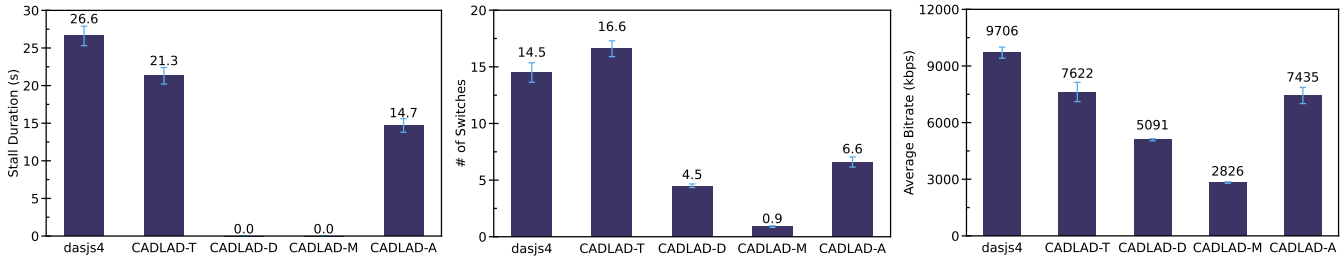
Similar to the results in VoD streaming, CADLAD is able to save a large amount of downloaded data. Each client using dashjs4 downloads segments with nearly $10\,000\,\mathrm{kbit/s}$ of average bitrate, whereas the clients in CADLAD-T use only $7622\,\mathrm{kbit/s}$ for every segment. CADLAD-D and CADLAD-M even download lower bitrates with $5091\,\mathrm{kbit/s}$ and $2826\,\mathrm{kbit/s}$ per segment, respectively, as shown in Fig. 3c.

The main target of CADLD is to improve the QoE of the clients which can be seen in Table III. We can see that CADLAD provides more QoE improvements when the devices are smaller. Dashjs4 clients have a QoE score of only 1.6 for the TV whereas CADLAD-T provides a 2.1 QoE score, which is a 31% improvement. When all of the clients are desktops, CADLAD-D achieves a more than 2.5 times higher QoE with a 4.3 QoE score, compared to 1.7 for dashjs4. In addition, CADLAD-M enhances the QoE by nearly 2.6 times from 1.76 of dashjs4. When a mix of devices joins a streaming session, CADLAD-A achieves a 2.4 QoE score, compared to 1.6 for dashjs4. These achievements come from short stall durations and a small number of quality switches of CADLAD while our proposed approach selects a suitable bitrate ladder based on the device type and the network conditions.

V. Conclusions

The CMCD specification provides useful information from the client to the server with the purpose of improving the performance of *HTTP Adaptive Streaming*.

In this paper, we address these questions by proposing a CMCD-aware per-device bitrate ladder construction, namely CADLAD. Our proposed approach provides the server with the top bitrate (`tb`) parameter according to the CMCD specification, the device type (`dt`) and the screen width (`sw`) of the device. The CADLAD calculates `tb` by the average value of the recent throughput to express the maximum bitrate that



(a) Stall duration. (b) Number of switches. (c) Average bitrate.

Fig. 3: Experimental results for multiple clients in live streaming with the Cascade network trace.

should be downloaded by the client. The server determines a suitable bitrate ladder for each client based on `tb`, `dt`, and `sw` from a list of bitrate levels then sends the corresponding MPD file to the client. The experimental results show that CADLAD is able to significantly improve the QoE for the end-user while saving substantial downloaded data to the client.

Acknowledgment

The financial support of the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association, is gratefully acknowledged. Christian Doppler Laboratory ATHENA: https://athena.itec.aau.at/, and in part by Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number T2EP20221-0023.

References

[1] Sandvine, "Global internet phenomena report 2022," [Online] Available: https://www.sandvine.com/phenomena, accessed: 02 July 2022.

[2] Bitmovin, "Video Developer Report 2021," [Online] Available: https://go.bitmovin.com/video-developer-report-2021, accessed: 02 July 2022.

[3] J. Stoll, "Devices used to watch online streaming videos in the United States as of the 4th quarter of 2020," [Online] Available: https://www.statista.com/statistics/784383/online-video-devices-in-the-us/, accessed: 02 July 2022.

[4] L. Ceci, "Devices used to watch online video worldwide as of August 2019," [Online] Available: https://www.statista.com/statistics/784351/online-video-devices/, accessed: 02 July 2022.

[5] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. D. Cock, "VMAF: The Journey Continues," [Online] Available: https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12, accessed: 02 July 2022.

[6] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 562–585, 2019 (DOI: 10.1109/COMST.2018.2862938).

[7] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, 2014, https://doi.org/10.1109/JSAC.2014.140403.

[8] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 187–198, https://doi.org/10.1145/2619239.2626296.

[9] Consumer Technology Association, "CTA-5004: Web Application Video Ecosystem - Common Media Client Data," [Online] Available: https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5004-final.pdf, accessed: 02 May 2022.

[10] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, "Common media client data (cmcd) initial findings," in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2021, pp. 25–33.

[11] A. C. Begen, A. Bentaleb, D. Silhavy, S. Pham, R. Zimmermann, and W. Law, "Road to salvation: Streaming clients and content delivery networks working together," *IEEE Communications Magazine*, vol. 59, no. 11, pp. 123–128, 2021.

[12] Apple, "HLS Authoring Specification for Apple Devices," [Online] Available: https://developer.apple.com/documentation/http_live_streaming/http_live_streaming_hls_authoring_specification_for_apple_devices, accessed: 02 July 2022.

[13] Google, "Recommended upload encoding settings," [Online] Available: https://support.google.com/youtube/answer/1722171, accessed: 02 July 2022.

[14] Twitch, "Broadcasting Guidelines," [Online] Available: https://stream.twitch.tv/encoding/, accessed: 02 July 2022.

[15] A. E. Al-Issa, A. Bentaleb, T. Zinner, I.-H. Mkwawa, and B. Ghita, "BBGDASH: A Max-Min Bounded Bitrate Guidance for SDN Enabled Adaptive Video Streaming," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2019, pp. 307–314.

[16] V. P. K. M, C. Timmerer, and H. Hellwagner, "MIPSO: Multi-Period Per-Scene Optimization For HTTP Adaptive Streaming," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[17] J. De Cock, Z. Li, M. Manohara, and A. Aaron, "Complexity-based consistent-quality encoding in the cloud," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1484–1488.

[18] V. V. Menon, H. Amirpour, M. Ghanbari, and C. Timmerer, "Opte: Online per-title encoding for live video streaming," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 1865–1869.

[19] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[20] B. Taraghi, A. Zabrovskiy, C. Timmerer, and H. Hellwagner, "CAdViSE: cloud-based adaptive video streaming evaluation framework for the automated testing of media players," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 349–352.

[21] B. Taraghi, H. Amirpour, and C. Timmerer, "Multi-codec ultra high definition 8k mpeg-dash dataset," in *Proceedings of the 13th ACM Multimedia Systems Conference*, 2022.

[22] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4G LTE dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 460–465.

[23] ITU-T, *Rec. P.1203. Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport - video quality estimation module*, Std., accessed: 02 July 2022. [Online]. Available: http://handle.itu.int/11.1002/ps/P1203-01

[24] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M. N. Garcia, and K. Yamagishi, "HTTP adaptive streaming QoE estimation with ITU-T rec. P. 1203: open databases and software," in *Proceedings of the 9th ACM Multimedia Systems Conf.*, 2018, pp. 466–471, https://doi.org/10.1145/3204949.3208124.