An Innovative Adaptive Web-Based Solution for Improved Remote Co-Creation and Delivery of Artistic Performances

Mohammed Amine Togou, *Member, IEEE*, Anderson Augusto Simiscuka, *Member, IEEE*, Rohit Verma, *Member, IEEE*, Noel E. O'Connor, *Member, IEEE*, Iñigo Tamayo, Stefano Masneri, Mikel Zorrilla, and Gabriel-Miro Muntean, *Fellow, IEEE*

***Abstract*—Due to the COVID-19 pandemic, most arts and cultural activities have moved online. This has contributed to the surge in development of artistic tools that enable professional artists to produce engaging and immersive shows remotely. This article introduces *TRACTION Co-Creation Stage (TCS)*, a novel Web-based solution, designed and developed in the context of the EU Horizon 2020 TRACTION project, which allows for remote creation and delivery of artistic shows. TCS supports multiple artists performing simultaneously, either live or pre-recorded, on multiple stages at different geographical locations. It employs a client-server approach. The client has two major components: Control and Display. The former is used by the production teams to create shows by specifying layouts, scenes, and media sources to be included. The latter is used by viewers to watch the various shows. To ensure viewers' good quality of experience (QoE) levels, TCS employs adaptive streaming based on a novel Prioritised Adaptation solution based on the DASH standard for pre-recorded content delivery (PADA), which is introduced in this paper. User tests and experiments are carried out to evaluate the performance of TCS' Control and Display applications and that of PADA algorithm when creating and distributing opera shows.**

***Index Terms*—Prioritised adaptive multimedia streaming, live and on-demand media delivery, quality of experience.**

I. Introduction

CO-CREATION in arts focuses on bringing individuals, communities, and professional artists together to create artistic works that capture their personality and political views as well as their life experiences and struggles. Co-Creation has been used as a platform to promote health and well-being for old people [1], including people with dementia [2]. It has also been used to address societal issues such as inequality [3], marginalisation [4], poverty [5], migrant integration [6], urban planning [7], and knowledge mobilisation [8].

TRACTION [9] is an European project that promotes co-creation to combat the problem of social exclusion in European societies. It aims at empowering marginalised communities by developing an effective, collaborative, and participatory production workflow for the creation of art representations, opera in particular, using new technologies. Three exploratory operas involving three distinct communities have been agreed on: *The Lost Cat*, a community opera in which 300 local residents of the Raval neighborhood in Barcelona, Spain would participate; *Time (As We Are)*, a community opera involving professional artists and a inmates from a youth prison community in Leiria, Portugal; and *Out of the Ordinary*, the world's first opera in virtual reality to be co-created by Irish National Opera and rural communities in Ireland.

During the COVID-19 pandemic, art venues were shut down completely for several months and rehearsals were prohibited in person [10]. As a result, several art activities, including TRACTION's, have moved online. This situation has contributed to the growth of not only art that has been digitised, but also art that has been created digitally. Subsequently, several cloud-based tools have been developed or customised to enable artists to remotely collaborate and produce vibrant art pieces. These tools deploy various delivery adaptation techniques [11], [12], [13], [14], [15], [16], [17], [18] to cope with the intrinsic characteristics of today's networks (*i.e.,* limited and highly volatile bandwidth), ensuring therefore a satisfactory quality of experience (QoE) to their users [19], [20], [21], [22]. Despite their ingenuity, these tools along with other existing projects experience three major issues. First, most of them lack the support for very high quality audio and video. Second, almost all of them do not jointly support on-demand and live media content. Finally, they deploy delivery adaptation techniques that adjust the quality of all media streams uniformly, without considering the importance of each stream to the overall artistic performance.

The main contributions of this paper are in terms of design and implementation of the following:

Manuscript received 13 December 2023; revised 27 January 2024; accepted 1 February 2024. Date of publication 13 March 2024; date of current version 7 June 2024. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme through the TRACTION Project under Grant 870610, and in part by the Science Foundation Ireland under Grant 12/RC/2289_P2 (Insight) and Grant 21/FFP-P/10244 (FRADIS). *(Corresponding author: Gabriel-Miro Muntean.)*

Mohammed Amine Togou is with the School of Computing, Dublin City University, Dublin 9, D09 DD7R Ireland (e-mail: mohammedamine.togou@dcu.ie).

Anderson Augusto Simiscuka, Noel E. O'Connor, and Gabriel-Miro Muntean are with the Insight SFI Centre for Data Analytics and the School of Electronic Engineering, Dublin City University, Dublin 9, D09 DD7R Ireland (e-mail: gabriel.muntean@dcu.ie).

Rohit Verma is with the School of Computing, National College Ireland, Dublin 1, D01 K6W2 Ireland.

Iñigo Tamayo and Mikel Zorrilla are with the Digital Media Department, Vicomtech, 20009 Donostia-San Sebastian, Spain.

Stefano Masneri is with the Computer Languages and Systems Department, University of the Basque Country UPV/EHU, 20018 Donostia-San Sebastian, Spain.

Digital Object Identifier 10.1109/TBC.2024.3363455

1) ***TRACTION Co-creation Stage (TCS)***, an innovative Web-based solution that supports both live and on-demand media to enable professional artists and individuals from the TRACTION's targeted communities to remotely collaborate to produce engaging and immersive opera shows, which might involve multiple simultaneous performances on distinct stages at different geographical locations. TCS adopts a client-server approach. The client has two major components, which focus on *Control* and *Display*, respectively. The former is used by the production teams to create the artistic shows while the latter is used to enable viewers to watch these shows on different devices (e.g., projectors, computers, smartphones).
2) ***P****rioritised* ***A****daptation algorithm based on the* ***DA****SH standard* ***(PADA)*** for streaming pre-recorded content which is fully integrated within the TCS client *Display* component. PADA aims at improving the viewers' QoE through reducing bitrate switches while averting playback interruptions via assigning priorities to the various on-demand streams considering their significance to the overall opera show. The most suitable bitrate for each stream is then selected based on a variety of factors, i.e., content's priority, available bandwidth, playback time, and quality variation.

The rest of this paper is organised as follows. First, we outline the approach used to gather the design requirements for TCS and survey existing commercial tools and projects. Then, we illustrate the architecture of TCS, describe its components, and shed light on the implementation part of the various elements as well as the challenges encountered. Afterwards, we outline user tests and the experiment study we run to assess the performance of the *Control* and *Display* components and discuss the results. Finally, we present our conclusions together with future research directions.

II. Related Work

A. Design Requirements and Existing Commercial Tools

A user-centric approach was deployed to gather requirements, with the goal of informing the design of the TRACTION Co-Creation Stage. Initial discussions among the project partners were held to identify the exploratory operas to use TCS along with potential users. The *Time (As We Are)* opera show was chosen unanimously as the most suitable to use TCS as it involves inmates who would not be allowed to travel to the opera house in Lisbon (Portugal) to perform alongside professional artists. Therefore, three user categories were determined: 1) professional artists, including production and co-creation teams; 2) non-professional artists, mainly the inmates in the youth prison; and 3) the audience, people who would be watching the opera show either live (on stages in Lisbon and Leiria) or remotely.

Following these discussions, a focus group and individual interviews were conducted by the technical team involving opera producers, professional artists, and people from the youth prison community. The goal of these activities was to understand the following: a) the current practices of opera productions; b) the show's objectives and its technological requirements; and c) how TCS is intended to be used during rehearsal/production phases to achieve these objectives. Data from the focus group and the individual interviews were then cleaned, transcribed, and analysed using open coding procedures. Five main design requirements for TCS were subsequently identified:

- *Orchestration:* the ability to manage multiple media sources simultaneously, supporting different types of media content, and enabling viewers to watch the show across a variety of devices.
- *Synchronisation:* enabling professional artists and the youth prison community to participate in the performance from different locations while ensuring synchronisation between the various sources.
- *Production functionalities:* providing capabilities such as timeline management and layout design (*i.e.,* number of sources, how they will look like) while supporting different source types (live, on-demand) and special effects.
- *Efficient delivery:* adaptive transmission of audio and video over the Internet to meet network bandwidth constraints while ensuring good QoE.
- *Universality:* professional artists and individuals from the youth prison community should be able to use the solution on any device to participate in a show. The audience should also be able to watch the show on any device.

There exist commercial tools that implement one or several of these requirements. For instance, *Jamulus* and *Jamkazam* enable synchronised online music playing. *Parse* and *Firebase* provide platforms that ensure synchronisation among WebRTC streams. *Abelton Live*, *OBS Studio* and *Max MSP 8* are music production tools. *Carbyne* and *BVMS* are cloud-based communication platforms that support orchestration while *OBS Ninja* is a Web-based video conferencing tool implementing synchronisation. There are also research projects that focused on designing multi-screen entertainment solutions. 2-IMMERSE [23] is an open-source platform for multi-screen entertainment services with content customisability based on device type, bandwidth, and viewer preferences. MediaScape [24] is a framework that enables the creation and distribution of Web-based media services over multiple devices. VConect [25] is an initiative to enable a theatre performance distributed over two stages at different locations. Orchestra [26] is an online platform that allows musicians at different geographical locations to perform together. Finally, LOLA [27] is a low latency audio video streaming system that enables real-time musical performances where musicians are physically located in remote sites.

While they provide some of the required features, these tools and projects do not support the full list of requirements listed above. Hence, the TRACTION Co-Creation Stage has been designed to meet all the specified requirements (*i.e.,* as illustrated in Fig. 1) while ensuring high quality content to help professional artists and the targeted communities create and run collaborative performances.

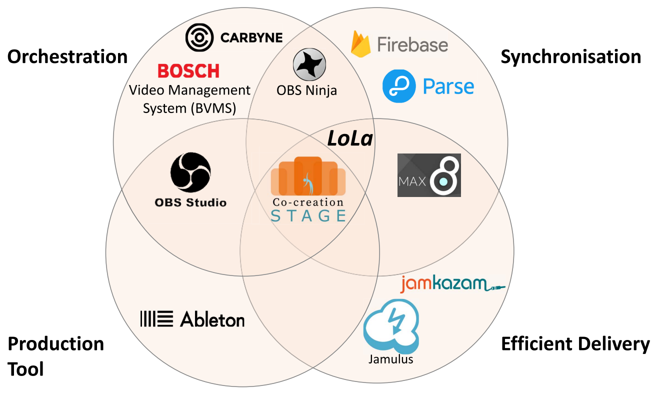


Fig. 1. TCS vs. existing commercial tools.

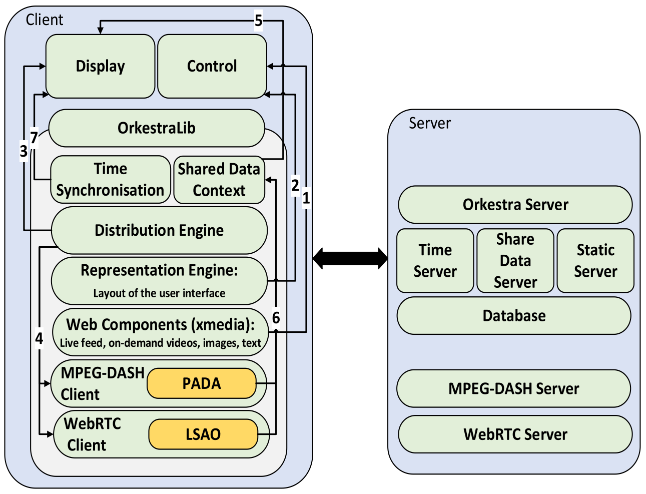


Fig. 2. Co-Creation Stage block architecture.

B. DASH-Based Adaptation Solutions

Numerous DASH-based adaptation algorithms have been proposed in the literature. Spiteri et al. [28] formulated the video bitrate adaptation as a utility maximization problem and proposed BOLA, an online algorithm that uses Lyapunov optimization techniques, to select the bitrate for next segments based solely on the amount of data in the buffer. The authors also proposed DYNAMIC [29] that uses bandwidth estimation when the buffer level is low and switches to BOLA when the buffer level is high to minimise rebuffering and bitrate oscillations while maximising the average video bitrate. Yaqoob et al. [16] proposed TBOA, a throughput and buffer occupancy-based adaptation scheme which downloads the first few segments with the lowest bitrate and adjusts the bitrates of the subsequent segments based on bandwidth estimations and buffer level. Zhou et al. [30] proposed a Markov decision-based rate adaptation scheme that takes into account video playback quality, bitrate switching frequency and amplitude, buffer level and buffer underflow/overflow events.

Sani et al. [31] proposed a supervised machine learning approach that takes as input the output of nine ABR algorithms across various streaming scenarios and predicts the optimal bitrate to be used for the next segment to be downloaded. Kim and Chung [32] proposed a reinforcement learning-based scheme for multi-client adaptive streaming which uses mobile edge servers to collect clients' information and feed them to a neural network model to select the optimal bitrate for the segments to be requested. Huang et al. [33] proposed an ABR algorithm that combines a deep reinforcement learning approach with a traditional buffer-based method to select the segments' optimal bitrate. The reinforcement learning method takes network status, video features, and QoE metrics as input and generates a buffer-bound value to control the buffer-based technique with the goal of maximising QoE. Armijo et al. [34] combined machine learning techniques with an edge-based ABR mechanism to improve the QoE by managing trade-off between bitrate, bitrate oscillations, and stalls according to network conditions. Finally, Li et al. [35] proposed RAV, a deep reinforcement learning ABR scheme that selects bitrates for each chunk's audio and video to guarantee high playback quality, avoid frequent stalls, and mitigate bitrate oscillations.

While these schemes demonstrate good performances, they may not be suitable for multi-stream scenarios since they might adapt all the streams equally, regardless of their importance. In addition, they give precedence to video over audio when performing adaptation, which may impact the viewers' QoE when watching opera shows. Therefore, we propose PADA, an adaptation scheme that prioritises media flows and adjusts their bitrates considering available bandwidth, quality variation, and buffer level.

III. TRACTION CO-CREATION STAGE (TCS)

With the feedback from the focus group and the individual interviews, use cases of the TCS usage were developed. One of them is illustrated in Fig. 3. The orchestra is playing a symphony on Stage 1 equipped with three screens, denoted as *displays* in this paper. The first two displays show live feeds of an opera play taking place in Stage 2 and a musician playing guitar at home. The third display show some pre-recorded videos. In Stage 2, there is only one display showing the live feed of the orchestra in Stage 1. The audience of the show is made of people present in Stages 1 and 2 as well as remote users watching the show on their devices.

A. Overall Architecture

Taking into account the developed use cases along with the requirements discussed in Section II, the development team decided to design TCS as a Web-based application having a client-server architecture, as depicted in Fig. 2. One of the key modules of the client is ***OrkestraLib,***[1] a library that provides support for complex functions for component management, layout configuration, and distribution efficacy through techniques such as plugin-based systems, under coupling, and modular injection to enable multi-device and multi-user applications, *e.g.,* the *Control* and *Display*, described in the next subsections. It abstracts the complexity of synchronising multi-device communications and is compatible

[1]https://github.com/tv-vicomtech/orkestraClient

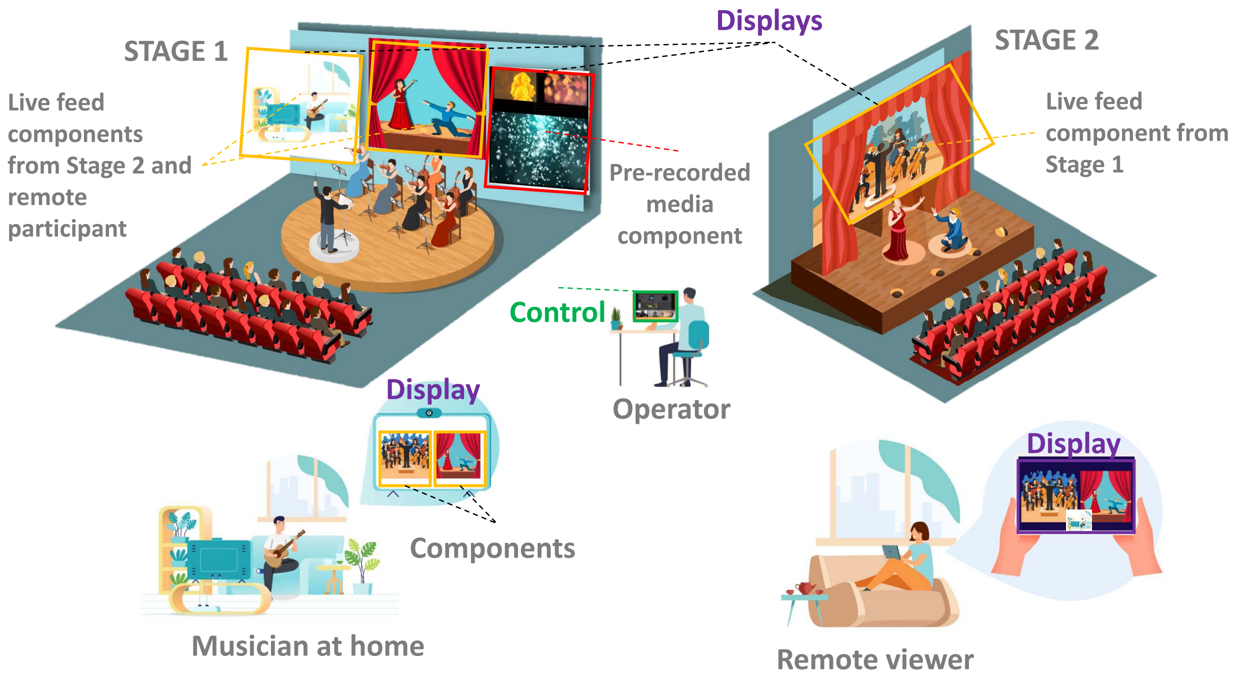

STAGE 1
Live feed components from Stage 2 and remote participant
Displays
Pre-recorded media component
STAGE 2
Live feed component from Stage 1
Control
Operator
Display
Components
Musician at home
Display
Remote viewer

Fig. 3. A use case of how TCS can be used to enable a distributed artistic show made of live and pre-recorded media sources from different locations.

with numerous frameworks such as Angular, React, es6 and vanilla. It integrates the following modules:

- *Web Components:* manages the resources to be used in the show, including live streams through WebRTC and on-demand videos using MPEG-DASH as well as images and text.
- *Representation Engine:* allows structuring the show's layout to display the Web components to be used. For instance, a layout can be structured to have multiple components having the same size or different sizes. The layout definition is done for each scene and for each display.
- *Distribution Engine:* takes care of the media distribution for real-time communication in both directions (*i.e.,* case of remote participants where they will broadcast their feed and see feeds from other stages) as well as for on-demand streams.
- *Time Synchronisation:* gets the time information from the time server to enable synchronisation across different displays (flow 7 in Fig. 2). For example, in case an on-demand video is shown on multiple displays, this module enables a frame-accurate synchronisation across all displays.
- *DASH and WebRTC Clients:* ensures adaptive delivery of on-demand and live media content. The former deploys the proposed PADA, while the latter integrates the **L**ive **S**tream **A**daptation algorithm for **O**pera (LSAO), which is not covered in this paper.
- *Shared Data Context:* keeps track of all the required data context from all displays involved in the show. As an example, if the operator moves from one scene to another, this information should be propagated to all displays (flow 5 in Fig. 2) so they know how to act (*e.g.,* change the components to be shown, use a different layout). It also contains the bitrates selected by PADA and LSAO for each media stream to enable bandwidth sharing between pre-recorded and live media (flow 6 in Fig. 2).

The server maintains the environment's consistency through the integration of the following modules:

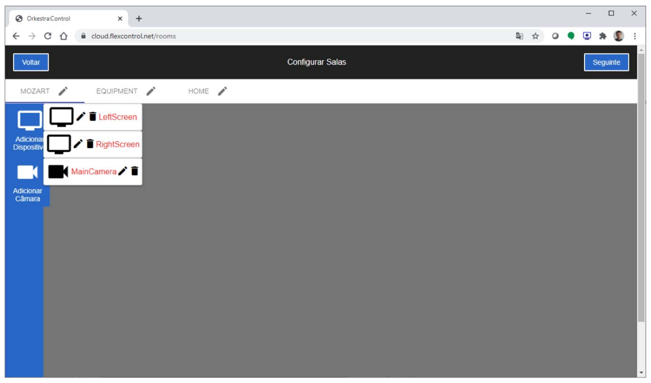
- *Orkestra Server:* organises and maintains multi-device sessions and the shared data coherently.
- *Time Server:* keeps track of timing information of various components and shares it with the ***Time Synchronisation*** module to enable synchronisation.
- *Shared Data Server:* enables the propagation of the Shared Data Context in real-time.
- *Static Server:* stores and loads templates (*e.g.,* HTML elements, static images, buttons) used to create shows.
- *MPEG-DASH and WebRTC Servers:* the former stores pre-recorded media components while the latter allows publishing and consuming live media feeds. Both can be deployed in the CTS server side or on the cloud.

Note that all modules in TCS were designed and developed by the technical team using JavaScript. The server-side was built on top of Node.js and MongoDB and is deployed in Amazon AWS server. The WebRTC server is based on JANUS. Both JANUS and MPEG-DASH server are deployed in Amazon AWS server.

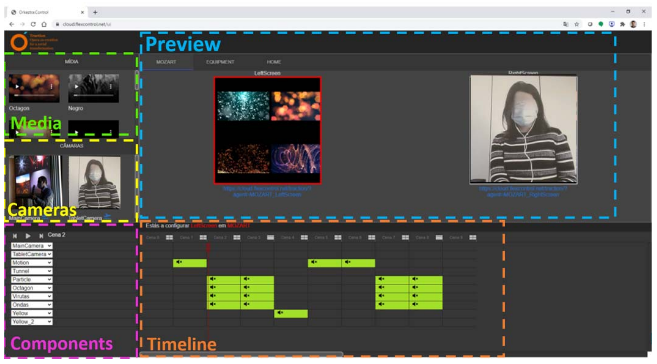
B. The Control Application

The *Control* component was instantiated by a *Control* application,[2] which was developed using the Angular Framework. It provides the interfaces for the creation of new shows. It is

[2]https://github.com/traction-project/CoCreationStage/tree/master/orkestra-control



(a) Interface 1



(b) Interface 2

Fig. 4. Interfaces of the Control application.

mainly used by an operator, following the lead of the show's director, to define the Web components to be used in the show (flow 1 in Fig. 2), when to use them (flow 2 in Fig. 2), and on which devices they will be shown. For instance, using the *Control* application, the operator in Fig. 3 defines three displays in Stage 1 and sets what would be depicted in each one of them: 2 for live streams and 1 for pre-recorded content.

Fig. 4 shows some of the interfaces of the ***Control*** application. Fig. 4(a) illustrates the interface to create rooms, define their names, and add input and display devices. In this example, the rooms are called MOZART, EQUIPMENT, and HOME. The MOZART room has two displays and one input device (*i.e.,* camera). Fig. 4(b) depicts the main interface for the ***Control*** application. The Media and Camera areas list all the on-demand and live components available for the show. If the live components are not connected, the "NO SIGNAL" icon will be shown. The Components area enables the operator to specify where each component will be displayed. In the Preview area, the operator can see a preview of what is happening in all the rooms by browsing the tabs of the different rooms. The preview windows depict exactly what the displays will be showing in a specific scene. the Timeline area is where the scenes are defined. Each column represents a scene where the operator specifies the components to be used for each one of them (the green boxes). The operator also sets the layout for each scene in each display.

C. The Display Application

The *Display* component was instantiated by *Display*,[3] which is a Web application built on top of JS Vanilla. It handles devices that work as displays during the show, *e.g.,* projectors in the main stages, cameras, and viewers' devices (computer monitors, mobile phones). It uses Web components following the Web Components standard[4] to enable operators to show/hide specific media (*i.e.,* pre-recorded and live content) on display devices based on the show layout (flow 3 in Fig. 2). A sample of the *Display* interface is depicted in Fig. 6. The *Display* application makes use of the adaptation algorithms provided by the Distribution Engine (flow 4 in Fig. 2) to adjust the quality of live and pre-recorded streams considering network conditions. One of these algorithms is PADA, which is described next.

[3]https://github.com/traction-project/CoCreationStage/tree/master/orkestraApp
[4]https://www.webcomponents.org/introduction

D. Prioritised Adaptation Based on DASH (PADA)

PADA is based on the work in [14] and has six modules, as illustrated in Fig. 5.

1) Layout Monitor (LM): assigns priority to on-demand components based on their importance to the overall show. Three priorities are considered: *high (H), medium (M), and low (L)*. The priority of each component is defined by the operator following the instructions of the show's director. LM then creates a table mapping each priority to a set of bitrates, *e.g.,* if the list of available video bitrates (Kbps) is $b = \{500, 1000, 1500, 2500, 4000\}$, *LM* uses b for priority H, $b - \{4000\}$ for priority M, and $b - \{4000, 2500\}$ for priority L. Note that in case of high network bandwidth (*e.g.,* > 1500Kbps), *LM* uses the full list of bitrates (b) for all priorities.

2) Bandwidth Estimator (BE): predicts the network's bandwidth using a smoothed moving average prediction approach as expressed in Eq. (1), where $\alpha_1, \alpha_2 > 0$ are smoothing coefficients. Note that Eq. (1) takes into consideration the short spikes and drops in bandwidth that may incur over time and which may influence the bandwidth estimations, yielding high bitrate variability.

$$BW_i^e = \begin{cases} BW_{i-1}, & i = 1 \\ \alpha_1 BW_{i-1} + \alpha_2 BW_{i-1}^e, & i > 1 \end{cases} \tag{1}$$

3) Playback Unit (PU): keeps track of the buffers' occupancy, governed by the time it takes to download new segments which is proportionally related to the selected bitrates. Let $B_i \in [0, B_{max}]$ be the buffer level after downloading segment i, with B_{max} denoting the maximum buffer size. We can express the dynamics of the buffer level as follows:

$$B_i = B_{i-1} + 1 - \left[\frac{d_i}{T}\right] \tag{2}$$

where d_i and T are the download time and the duration of segment i, respectively. T is set to be the same for all the segments. The first term in Eq. (2) represents the buffer level before downloading segment i while the second term

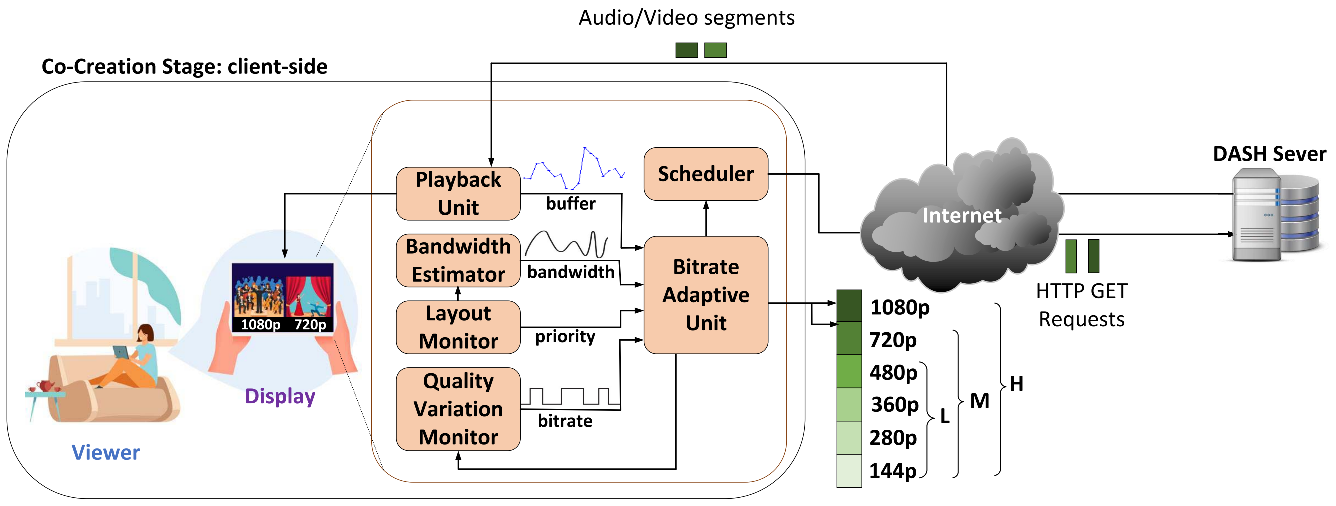


Fig. 5. PADA's block architecture and its deployment.

represents the increment of the buffer level once segment i is fully downloaded. The third term represents the number of segments played while downloading segment i. [.] is used for rounding the third term to the closest integer. By analyzing Eq. (2), we can observe that it is dominated by the third term. Indeed, when d_i is long, the playback buffer may run dry (*i.e.,* buffer underflow) before fully downloading segment i, leading to playback interruptions. In this case, a rebuffering period is triggered in which the buffer is filled while the streaming is paused. On the other hand, when d_i is short, segments will be continuously downloaded into the buffer, which might induce buffer overflow. To mitigate these problems, we introduce two thresholds: B_l and B_h. In case the buffer level is less than B_l, segments will be downloaded at low bitrates to quickly fill in the buffer, avoiding therefore streaming interruptions. In case the buffer level is greater than B_h, segments will be downloaded with higher bitrates to allow for the buffer level to be reduced, averting the buffer overflow problem.

4) Bitrate Adaptive Unit (BAU): selects the appropriate bitrate for each on-demand stream. Let b_{min} and b_{max} be the dynamic bounds of the bitrates list b provided by **LM** and let k be the number of segments. Initially, b_{min} and b_{max} are both set to the lowest bitrate in b as no bandwidth estimation exists (see Algorithm 1). After downloading few segments, the bounds are adjusted to reflect the estimated bandwidth. If it tends to increase, b_{max} is set to the highest bitrate that is lower than the estimated bandwidth while b_{min} is set to the next highest bitrate in the list (lines $9-13$). If it decreases, b_{min} is set to the bitrate having an index that is two decrements from the index of b_{max} (lines $14-18$). This is to limit the number of bitrates from which PADA should choose in order to avert substantial bitrate oscillations when sudden short bandwidth changes occur.

Algorithm 2 illustrates the bitrate selection process. PADA adopts a more liberal approach where the first segment is downloaded at the lowest supported bitrate (*i.e.,* due to missing bandwidth estimates) while the bitrate of the following segments is selected based on the estimated download time and the buffer level, *i.e.,* should not drop to 0. The same bitrate is

Algorithm 1: Dynamic Adjustment of b_{min} and b_{max}

```
Input: b - List of available bitrates
Result: $b_{min}$ and $b_{max}$
while $i \neq k$ do
    if $i = 1$ then
        $b_{min} \leftarrow min(b)$ and $b_{max} \leftarrow min(b)$
        $BW_{last} \leftarrow 0$
    else
        if $i > 2$ then
            $BW_{last} \leftarrow BW_{i-2}$
        end
        if $BW_{i-1} - BW_{last} > 0$ then
            if $b_{max} \leq BW_{i-1}$ then
                $b_{max} \leftarrow \max \{b_j \in b \mid b_j \leq BW_{i-1}\}$
                $b_{min} \leftarrow \min \{b_j \in b \mid b_j > b_{min}\}$
            end
        else
            if $b_{min} > BW_{i-1}$ then
                $b_{max} \leftarrow \max \{b_j \in b \mid b_j \leq BW_{i-1}\}$
                $b_{min} \leftarrow \max \{b_j \in b \mid max - j = 2\}$
            end
        end
    end
end
```

used for the following segments till the buffer level exceeds B_l. In case the buffer level is between B_l and B_h, PADA selects the bitrate that meets the conditions in Eq. (3). The first condition indicates that the bitrate of segment i for the stream with highest priority p should not exceed the estimated bandwidth. The second condition specifies that the bitrate of segment i for streams with priority M and L should not be higher than the bitrate of the stream with the highest priority (b_{i_H}). The third condition indicates that the selected bitrate should be the next bitrate in the list with respect to b_{i-1} in either ascending or descending order. The last condition implies that the buffer level should be higher than the threshold B_l after downloading

Algorithm 2: Bitrate Selection

```
Result: $b_i$, bitrate of segment $i$
while $i \neq k$ do
    if $i = 1$ then
        $b_i \leftarrow b_{min}$
    else
        if $B_{i-1} \leq B_l$ then
            $b_i \leftarrow \max\ \{b_j \in b \text{ s.t. } (T \times B_i) - d_i > 0\}$
        end
        if $B_l < B_{i-1} \leq B_h$ then
            $b_i \leftarrow \max\ \{b_j \in b \text{ s.t. Eq. (3) holds}\}$
        end
        if $B_{i-1} > B_h$ then
            wait for $\tau$ seconds
        end
    end
end
```

segment i.

$$\begin{cases} b_i \leq BW^e_{i-1} & \text{, if } p = H \\ b_i \leq BW^e_{i-1} - b_{i_H} & \text{, if } p = M, L \\ \dfrac{|b_i - b_{i-1}|}{q_i} \leq 1 & \\ (T \times B_i) - d_i \geq T \times B_l & \end{cases} \tag{3}$$

In case the buffer level exceeds B_h, PADA waits for a period of time before requesting the next segment. This is to avoid the buffer overflow problem. Still, chances of the buffer underflow occurring during this period cannot be ignored particularly in the case of a sharp drop in bandwidth or a low number of segments in the buffer once this period expires. As a result, the waiting period τ is computed as follows:

$$\tau = T\left(B_{i-1} - \left[\frac{B_l + B_h}{2}\right]\right) \tag{4}$$

5) Quality Variation Monitor (QVM): keeps track of the difference in bitrates among the segments that have already been downloaded. Studies have shown that high bitrate variation among segments can significantly decrease the user's QoE. Therefore, to reduce the frequent bitrate switches, we use a moving average approach to keep track of the bitrate variation, computed as follows:

$$q_i = \begin{cases} b_1, & i = 1 \\ (1 - \beta)(b_i - b_{i-1}) + \beta q_{i-1}, & i > 1 \end{cases} \tag{5}$$

where $\beta \in [0, 1]$ is a smoothing coefficient. Note that Eq. (5) captures the short-term bitrate variation and gives higher weight to bitrate changes of recent segments as they are more likely to influence the users' perceived QoE.

Finally, the **Scheduler** sends HTTP GET requests to the DASH server to download segments with the selected bitrates.

Note that PADA tries to maintain the same bitrate, if possible, since abrupt variations in quality, particularly when the bandwidth is low, can negatively affect viewers' QoE. Note also that selected bitrates for on-demand streams are shared using the ***Shared Data Context*** module to enable inter-adaptation among them.

E. Challenges

While implementing the TRACTION Co-creation Stage, we encountered three main challenges.

First, operators have many tasks at hand and they lack a clear vision of the overall show as everything is decided by the show's director. This can be particularly frustrating in cases where abrupt or last minute changes are to be made to the show. As a result, we are examining the possibility of creating a new role, called *director*, which will be in charge of defining a template on the **Control** app that tells the story of the show (*i.e.,* the scenes, the components in each scene, and when and where to show each component). This will facilitate the operators' work.

The second challenge is related to the universality requirement. We want TRACTION Co-Creation Stage to be used from any browser on any device. Yet, problems related to unpredicted network conditions along with users' capability to properly utilise the application may arise. In addition, audio and video quality will greatly depend on devices used, regardless of the performance of the media adaptation algorithms, which might impact the viewers' QoE.

Last, TRACTION Co-Creation Stage uses the *getUserMedia()* function provided by the *Stream API*. While this function is supported by almost all browsers, it is used mainly for video calls and is not suited for transmitting music. For instance, *echoCancellation* and *NoiseSupression* is used for *getUserMedia* audio streams in Chrome, which deteriorates music sounds. In addition, the *getUserMedia* function applies often audio normalisation, which might remove the nuances of a melody, making people unable to distinguish between a pianissimo and a fortissimo. To address this issue, we enabled TRACTION Co-Creation Stage to use Blackmagic cards and External Audio interfaces to capture good quality video and audio and to deliver them without deterioration. However, a technician is needed in each location.

IV. Performance Evaluation

In this section, we first describe the user tests run to assess the usability of TCS. Then, we discuss the experiment that took place to evaluate PADA's performance.

A. Control App User Tests

Formal user tests were run in collaboration with the *Time (As We Are)* opera show team to assess the *Control* app's usability. Two operators are responsible for creating the show; as a result, they were the main participants in our user tests. Since these tests were run in the prison premise (*i.e.,* as it contains one of the main stages of the show), we could not invite the production teams of the two other opera shows involved in the project to take part in the tests due to restrictions imposed by the prison authorities. The two participants were asked to complete 8 tasks, described in Table I, representing samples of tasks that an operator should perform before and during the show.

After completing each task, they were asked to rank the task difficulty on a seven-point Likert scale anchored by *very easy* and *very difficult* (see Tables II and III). Once all the tasks

TABLE I
TASKS DESCRIPTION

Task Number	Description
1	Check cameras and start displays
2	Check connection from remote user
3	Start the show
4	Navigate from one scene to another
5	Browse the scenes till the end of the show
6	Change layout of a display
7	Change layout of a display
8	Load an existing show template

TABLE II
SUMMARY OF TASK COMPLETION TIME AND TDR OF PARTICIPANT 1

Task	1	2	3	4	5	6	7	8
Start	14:38	14:40	14:42	14:47	14:49	14:52	14:56	15:12
Finish	14:40	14:42	14:46	14:49	14:52	14:55	15:09	15:16
TDR	1	1	3	1	1	1	5	1

TABLE III
SUMMARY OF TASK COMPLETION TIME AND TDR OF PARTICIPANT 2

Task	1	2	3	4	5	6	7	8
Start	15:41	15:44	15:45	15:48	16:00	16:08	16:09	16:21
Finish	15:44	15:45	15:48	15:59	16:08	16:09	16:19	16:25
TDR	1	1	1	1	1	1	2	3

were completed, participants were asked to fill in a usability questionnaire with several single-item constructs, including:

1) The TRACTION Co-Creation Stage is unnecessarily complex.
2) The TRACTION Co-Creation Stage is easy to use.
3) I would need the support of a technician to use the TRACTION Co-Creation Stage.
4) The various functions in the TRACTION Co-Creation Stage are well integrated.
5) I learnt to use the TRACTION Co-Creation Stage very quickly.
6) I felt very confident when using the TRACTION Co-Creation Stage.
7) I needed to learn a lot of aspects before using the TRACTION Co-Creation Stage.

The answers to these constructs were recorded on a five-point Likert scale anchored by *Strongly disagree* and *Strongly agree* (see Table IV). Finally, individual interviews were conducted to understand the difficulties and problems encountered by the participants when completing the tasks and get their suggestions of possible improvements.

Discussion: **Participant 1** indicated that overall, the interface is quite easy to use and very intuitive. However, she mentioned that task 7 was the most difficult (see Table II as it took her some time to understand how the interface behaves. She also mentioned that she had difficulties understanding how to go from one scene to another. Indeed, she did not think that it was necessary to click on the scene's name but rather to click anywhere on the timeline. Moreover, she had problems finding a component in the media list that was not already used in the timeline and suggested to improve this functionality. Finally, she pointed out that when replacing one component in a specific scene, that component is replaced in all the other scenes.

Participant 2 also indicated that in general, the interface is easy to use and friendly, but there is room for improvement. He found that tasks 7 and 8 are the most complex (see Table III) as it was not trivial for him to figure out the interface's behaviour. He also indicated that when adding a new component to a scene, this component is added to all the following scenes. In terms of improvements, he suggested that the vertical bar in the timeline should show the scene number as well as slowly moving from left to right to indicate the time passed since the beginning of a scene. He also recommended that instead of showing 4 components in the current layout icon (*i.e.,* used to switch from mosaic to split layout and vice versa), it would be better to have icons that match the number of components selected for each display to enable the operator to immediately know the way the components will be displayed.

Note that the problems raised by both participants have been investigated and fixed. In addition, the suggested improvements have been analysed and translated into system requirements that are being implemented in the new development branch of the TRACTION Co-Creation Stage.

B. Display App Performance Evaluation

We run an experiment involving 33 participants, aged between 20 and 70, to assess the performance of the *Display* app with the support of several adaptation algorithms. Connected from 11 countries across 4 continents, participants (see their demographics and network setting in Tables V - VII) were asked to use the *Display* app to stream and watch an opera play made of 3 pre-recorded clips, stored in a DASH server in Dublin (Ireland) and are shown on three different views (*i.e.,* displays) as illustrated in Fig. 6. The clips were taken from "Só Zerlina ou Cosi fan Tutte?", an opera play created by Sociedade Artistica Muscial de Pousos-Portugal (SAMP), and have scenes with distinct characteristics (*i.e.,* dim vs. bright lights, group of people vs. individuals, dialog vs. singing, movement vs. stillness) to portray spatial and temporal variations. They are 2 minutes long with a frame rate of 25fps and are encoded using the H.264 encoder to provide high-quality transmission of videos in limited network bandwidth scenarios. Five video bitrates were used: 500Kbps (240p), 1000Kpbs (360p), 1500Kbps (480p), 2500Kbps (720p), and 4000Kbps (1080p). To enable multi-bitrate audio while supporting seamless switching between the various qualities, we used the xHE-AAC codec with 3 bitrates (192Kbps, 300Kbps and 600Kbps) with a sample rate of 48KHz. Clip 1 has priority *H* and is displayed in View 1, Clip 2 has priority *M* and is displayed in View 2, and Clip 3 has priority *L* and is displayed in View 3 (see Fig. 6).

After watching the opera show, participants were invited to fill in a QoE questionnaire with several single-item constructs for each clip, including:

1) Please rate the audio quality.
2) Please rate the video quality.
3) Have you experienced audio glitches?
4) Have you experienced video stalls?
5) Please rate your overall enjoyment of the experience.

TABLE IV
ANSWERS TO USABILITY QUESTIONNAIRE OF BOTH PARTICIPANTS

Participants	Construct 1	Construct 2	Construct 3	Construct 4	Construct 5	Construct 6	Construct 7
1	1	5	4	5	5	5	1
2	1	4	1	4	5	5	1

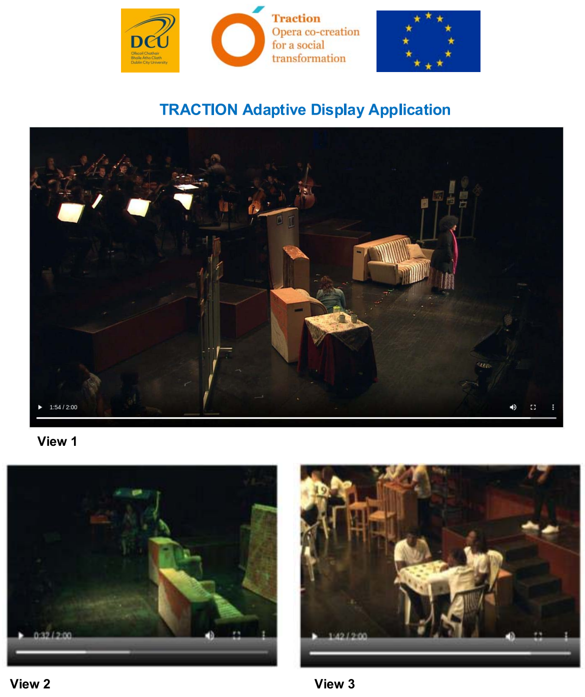

Traction
Opera co-creation for a social transformation
TRACTION Adaptive Display Application
View 1
View 2
View 3

Fig. 6. The Display App showing the opera show used in our experiment which is made of 3 pre-recorded clips.

TABLE V
PARTICIPANTS' AGE DISTRIBUTION

Age	Number of Participants
20 - 29	12
30 - 39	13
40 - 49	4
$\geq$ 50	4

TABLE VI
PARTICIPANTS' GENDER DISTRIBUTION

Gender	Number of Participants
Female	13
Male	20

TABLE VII
PARTICIPANTS' NETWORK DISTRIBUTION

Network Type	Number of Participants
WiFi	22
3G/4G/5g	4
Ethernet	7

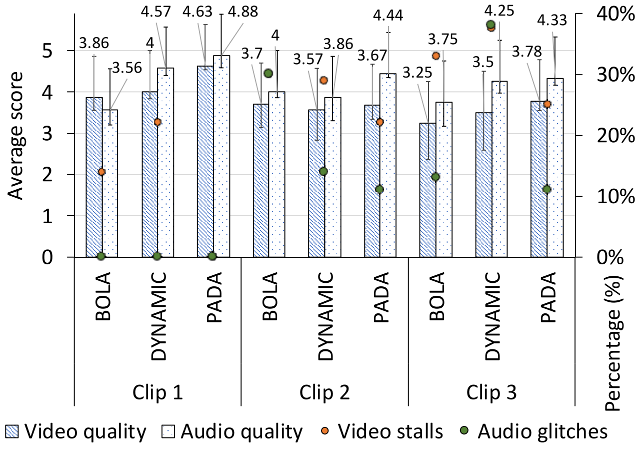


Fig. 7. Average scores of QoE metrics across all clips.

Answers to constructs 1,2, and 5 were recorded on a five-point Likert scale anchored by *poor* and *excellent*. Constructs 3 and 4 have "Yes/No" answers. Apart from the qualitative data, we configured the *Display* app to collect QoS metrics every second and send them to the server, including selected audio and video bitrates, bitrate switches, and time to reach the highest bitrate.

The segment duration, the maximum buffer size, B_l, and B_h were set to 2, 25, 10, and 20 seconds, respectively. Along with PADA, two commercially used adaptation algorithms were deployed: BOLA [28] and DYNAMIC [29]. Note that participants were not aware of which adaptation algorithm was used to eliminate any possible bias.

Fig. 7 illustrates the average scores of audio and video qualities (constructs 1 and 2) perceived by the participants. All results are reported with 95% confidence interval. We observe that PADA scored the highest in all clips in terms of audio quality. As for video quality, PADA got the highest score in clips 1 and 3, and second highest in clip 2. For instance, PADA's scores for clip 1 are higher than those of BOLA and DYNAMIC by 31% and 7% in terms of audio quality, and 18% and 15% in terms of video quality.

Fig. 7 also depicts the percentage of participants experiencing audio glitches and video stalls (constructs 3 and 4) across all clips. Again, all results are reported with 95% confidence interval. We observe that PADA incurred the lowest percentage of audio glitches and video stalls across all clips. While the three schemes did not incur any audio glitches in clip 1, PADA incurred audio glitches that are 19% and 3% less than those of BOLA and DYNAMIC for clip 2, and 2% and 27% less for clip 3. PADA also incurred no video stalls in clip 1 against 14% and 22% for BOLA and DYNAMIC, 8% and 7% less for clip 2, and 8% and 13% less for clip 3.

Table VIII depicts the collected QoS metrics across all clips. We observe that:

- PADA provides the highest audio and video average bitrate for clip 1, reflecting the high scores in Fig. 7, *i.e.,* PADA's average bitrates are higher than those of BOLA and DYNAMIC by 77% and 61% for audio, and 32% and 36% for video.

TABLE VIII
QoS Measurements Across All Clips

	Clip 1			Clip 2			Clip 3		
	BOLA	**DYNAMIC**	**PADA**	**BOLA**	**DYNAMIC**	**PADA**	**BOLA**	**DYNAMIC**	**PADA**
Average video bitrate (Kbps)	2782	2654	3829	3384	3371	3289	2863	3702	2938
Average audio bitrate (Kbps)	262	317	596	252	261	305	304	210	293
Average video bitrate switches	8	6	3	6	4	3	4	4	6
Average time to highest video bitrate (s)	13	46	26	0	18	30	30	25	48

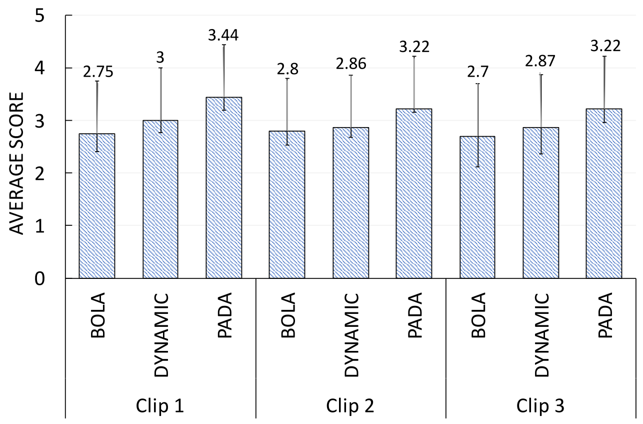


Fig. 8. Average score of participants' enjoyment in each clip.

- PADA provides the highest audio average bitrate for clip 2, reflecting the high score of audio quality in Fig. 7, *i.e.,* PADA's average bitrate is higher than those of BOLA and DYNAMIC by 19% and 16%. Note that even though PADA provides the lowest video bitrate in clip 2, it scores almost as high as BOLA in Fig. 7 as it incurred the lowest percentage of video stalls, known to affect the viewers' QoE.
- PADA incurred the least number of bitrate switches for clips 1 and 2 for video.
- PADA requires extended time to reach the highest video bitrate across all clips when compared to BOLA and DYNAMIC.

Finally, Fig. 8 illustrates the average score of participants' enjoyment across all clips. We observe that PADA scored the highest across the three clips. Indeed, the average enjoyment score incurred by PADA exceeded those of BOLA and DYNAMIC by 22% and 14% in clip 1, 14% and 12% in clip 2, and 18% and 11% in clip 3. This goes inline with the results in Fig. 7 as PADA strives to select high audio and video bitrates to match the network's bandwidth while reducing bitrate switches and ensuring low video stalls and audio glitches.

The reason behind PADA's performance is threefold. First, PADA selects the highest sustainable bitrate when downloading video segments. Second, PADA reacts more conservatively to changes in bandwidth to reduce the number of bitrate oscillations. This implies that PADA would require longer time to reach high bitrates, but only in case of low/moderate bandwidth. Third, by considering the quality variation with respect to previously downloaded segments, PADA averts high amplitude changes in bitrates. As a result, PADA provides a smoother streaming experience.

Note that when streaming both live and pre-recorded media, TCS will give precedence to live components as they have short buffers (*i.e.,* 3 to 8 seconds) and are of high significance to the overall performance. Consequently, PADA uses the information in the *Shared Data Context* when estimating the bandwidth to select the bitrate of pre-recorded components, which might be low in case of moderate bandwidth.

V. Conclusion and Future Works

This article describes TRACTION Co-Creation Stage (TCS), a Web-based solution that enables the creation and delivery of collaborative opera shows via *Control* and *Display* apps. The paper introduces a novel prioritised adaptive streaming scheme for pre-recorded content (PADA). Testing with on-demand opera content shows how when using PADA, the *Display* app provides higher enjoyment by ensuring reduced bitrate fluctuations, very important for achieving high overall viewer QoE.

We believe that TCS is a fundamental leap forward in the online entertainment business as it enables content delivery providers to flexibly adapt to the raise in demand in multi-source streaming content. We also believe that TCS can be useful in other business models such as those that rely on adapting camera feeds for surveillance purposes.

Future research will focus on running more user tests involving participants with different roles (*e.g.,* operators, professional and non-professional artists). We also plan to fully develop and test the live adaptation algorithm to enable the creation and delivery of live shows using TCS. Moreover, evaluating inter-stream bitrate adaption between live and on-demand components to further improve the viewers' QoE should be realised. Finally, performance optimization of TCS in terms of efficiency and scalability can be evaluated through use cases with varying number of displays and viewers.

References

[1] A. Terkelsen, C. Wester, G. Gulis, J. Jespersen, and P. Andersen, "Co-creation of activities to promote health and well-being of older people—A scoping review," *Eur. J. Publ. Health*, vol. 32, Oct. 2022, Art. no. ckac129.279.

[2] H. Zeilig, J. West, and M. van der Byl Williams, "Co-creativity: Possibilities for using the arts with people with a dementia," *Qual. Ageing Older Adults*, vol. 19, no. 2, pp. 135–145, 2018.

[3] J. Carpenter, C. Horvath, and B. Spencer, "Co-creation as an agonistic practice in the Favela of Santa Marta, Rio de Janeiro," *Urban Stud.*, vol. 58, no. 9, pp. 1906–1923, 2021.

[4] C. Horvath and J. Carpenter, *Co-Creation in Theory and Practice: Exploring Creativity in the Global North and South*. Bristol, U.K.: Policy Press, Sep. 2020. [Online]. Available: https://doi.org/10.1332/policypress/9781447353959.001.0001

[5] T. Nahi, "Co-creation for sustainable development: The bounds of NGO contributions to inclusive business," *Bus. Strat. Develop.*, vol. 1, no. 2, pp. 88–102, 2018.

[6] T. C. Turin, N. Chowdhury, S. Haque, N. Rumana, N. Rahman, and M. A. A. Lasker, "Involving im/migrant community members for knowledge co-creation: The greater the desired involvement, the greater the need for capacity building," *BMJ Glob. Health*, vol. 6, no. 12, 2021, Art. no. e007602.

[7] H. Leino and E. Puumala, "What can co-creation do for the citizens? applying co-creation for the promotion of participation in cities," *Environ. Plann. C Polit. Space*, vol. 39, no. 4, pp. 781–799, 2021.

[8] S. MacGregor, A. Cooper, M. Searle, and T. Kukkonen, "Co-production and arts-informed inquiry as creative power for knowledge mobilisation," *Evidence Policy*, vol. 18, no. 2, pp. 206–235, 2022.

[9] TRACTION Project, San Francisco, CA, USA. "Opera co-creation for social transformation." Jan. 2023. [Online]. Available: https://www.traction-project.eu/

[10] N. Spiro et al., "The effects of COVID-19 lockdown 1.0 on working patterns, income, and wellbeing among performing arts professionals in the United Kingdom (April–June 2020)," *Front. Psychol.*, vol. 11, p. 4105, Feb. 2021.

[11] L. Zhong, M. Wang, C. Xu, S. Yang, and G.-M. Muntean, "Decentralized optimization for multicast adaptive video streaming in edge cache-assisted networks," *IEEE Trans. Broadcast.*, vol. 69, no. 3, pp. 812–822, Sep. 2023.

[12] G. Zhou, Z. Luo, M. Hu, and D. Wu, "PreSR: Neural-enhanced adaptive streaming of VBR-encoded videos with selective prefetching," *IEEE Trans. Broadcast.*, vol. 69, no. 1, pp. 49–61, Mar. 2023.

[13] J. Fu, Z. Chen, X. Chen, and W. Li, "Sequential reinforced 360-degree video adaptive streaming with cross-user attentive network," *IEEE Trans. Broadcast.*, vol. 67, no. 2, pp. 383–394, Jun. 2021.

[14] M. A. Togou and G.-M. Muntean, "An elastic DASH-based bitrate adaptation scheme for smooth on-demand video streaming," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, 2022, pp. 1–6.

[15] Z. Ye et al., "VRCT: A viewport reconstruction-based 360° video caching solution for tile-adaptive streaming," *IEEE Trans. Broadcast.*, vol. 69, no. 3, pp. 691–703, Sep. 2023.

[16] A. Yaqoob, T. Bi, and G. Muntean, "A DASH-based efficient throughput and buffer occupancy-based adaptation algorithm for smooth multimedia streaming," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2019, pp. 643–649.

[17] D. Yun and K. Chung, "DASH-based multi-view video streaming system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 1974–1980, Aug. 2018.

[18] S. Q. Jabbar, D. J. Kadhim, and Y. Li, "Proposed an adaptive bitrate algorithm based on measuring bandwidth and video buffer occupancy for providing smoothly video streaming," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 1–10, 2018.

[19] D. Anand, M. A. Togou, and G.-M. Muntean, "A machine learning solution for video delivery to mitigate co-tier interference in 5G HetNets," *IEEE Trans. Multimedia*, vol. 25, pp. 5117–5129, 2022.

[20] P. Szabo, A. Simiscuka, S. Masneri, M. Zorrilla, and G.-M. Muntean, "A CNN-based framework for enhancing 360° VR experiences with multisensorial effects," *IEEE Trans. Multimedia*, vol. 25, pp. 3245–3258, 2022.

[21] T. Bi, R. Lyons, G. Fox, and G.-M. Muntean, "Improving student learning satisfaction by using an innovative DASH-based multiple sensorial media delivery solution," *IEEE Trans. Multimedia*, vol. 23, pp. 3494–3505, 2021.

[22] Q. Li et al., "A super-resolution flexible video coding solution for improving live streaming quality," *IEEE Trans. Multimedia*, vol. 253, pp. 6341–6355, 2022.

[23] J. Walker et al., "2-IMMERSE: A platform for production, delivery, and orchestration of distributed media applications," *SMPTE Motion Imag. J.*, vol. 128, no. 7, pp. 45–51, Aug. 2019.

[24] A. Domínguez, M. Agirre, J. Flörez, A. Lafuente, I. Tamayo, and M. Zorrilla, "Deployment of a hybrid broadcast-Internet multi-device service for a live TV programme," *IEEE Trans. Broadcast.*, vol. 64, no. 1, pp. 153–163, Mar. 2018.

[25] D. L. Williams et al., "A distributed theatre experiment with shakespeare," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 281–290.

[26] M. Rofe, S. Murray, and W. Parker, "Online orchestra: Connecting remote communities through music," *J. Music Technol. Educ.*, vol. 10, no. 3, pp. 147–165, 2017.

[27] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via LOLA: Low latency high quality A/V streaming system," in *Information Technologies for Performing Arts, Media Access, and Entertainment*, P. Nesi and R. Santucci, Eds. Berlin, Germany: Springer, 2013, pp. 240–250.

[28] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1698–1711, Aug. 2020.

[29] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the DASH reference player," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 2s, p. 67, Jul. 2019.

[30] C. Zhou, C. Lin, and Z. Guo, "mDASH: A Markov decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 738–751, Apr. 2016.

[31] Y. Sani, D. Raca, J. J. Quinlan, and C. J. Sreenan, "SMASH: A supervised machine learning approach to adaptive video streaming over HTTP," in *Proc. 12th Int. Conf. Qual. Multimedia Exp. (QoMEX)*, 2020, pp. 1–6.

[32] M. Kim and K. Chung, "Reinforcement learning-based adaptive streaming scheme with edge computing assistance," *Sensors*, vol. 22, no. 6, p. 2171, 2022.

[33] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1967–1976.

[34] J. A. Armijo, E. Çetinkaya, C. Timmerer, and H. Hellwagner, "ECAS-ML: Edge computing assisted adaptation scheme with machine learning for HTTP adaptive streaming," 2022, *arXiv:2201.04488*.

[35] W. Li, J. Huang, W. Lyu, B. Guo, W. Jiang, and J. Wang, "RAV: Learning-based adaptive streaming to coordinate the audio and video bitrate selections," *IEEE Trans. Multimedia*, vol. 25, pp. 5662–5675, 2022.

Mohammed Amine Togou (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science and computer networks from Al Akhawayn University in Ifrane, Morocco, and the Ph.D. degree in computer science from the University of Montreal, Canada. He is an Assistant Professor with the School of Computing, Dublin City University, Ireland. He has published over 40 peer-reviewed scientific articles in top journals and flagship conferences. He has also served as a member of the technical programme committee of several international conferences. His current research interests include 5G/6G networks, blockchain, machine learning, adaptive multimedia delivery, IoT, and technology enhanced learning.

Anderson Augusto Simiscuka (Member, IEEE) received the B.Sc. degree in information systems from Mackenzie Presbyterian University, São Paulo, Brazil, in 2014, and the Ph.D. degree from the School of Electronic Engineering, Dublin City University, where he is a Postdoctoral Researcher with the Performance Engineering Laboratory. His research is mainly focused on the Internet of Things communications performance, virtual reality, multisensorial media, and content adaptation. He is a member of the IEEE Communications Society and IEEE Broadcast Technology Society.

Rohit Verma (Member, IEEE) received the B.Eng. and M.Tech. degrees in computer science and information security from the Manipal Institute of Technology, Mahe, India. He is an Assistant Professor with the School of Computing, National College of Ireland, Dublin. He was awarded a Ph.D. degree by the Indian Institute of Technology Indore, India. Previously, he was a Postdoctoral Researcher with the Performance Engineering Laboratory, Dublin City University, Ireland. His current research areas include service computing, blockchain, adaptive systems, cybersecurity, and autonomous computing.

Noel E. O'Connor (Member, IEEE) is a Full Professor with the School of Electronic Engineering, Dublin City University, Ireland, and the CEO of the Insight SFI Research Centre for Data Analytics. His research focuses on multimedia content analysis, computer vision, machine learning, information fusion, and multimodal analysis for diverse applications. He is an Area Editor for *Signal Processing: Image Communication* (Elsevier) and an Associate Editor for the Journal of Image and Video Processing (Springer).

Iñigo Tamayo received the degree in computer science engineering from the University of Mondragon, Spain, in 2007, and the degree in computational engineering and intelligent systems from the University of Basque Country, Spain, in 2017. He is a Senior Software Developer with the Department of Digital Media, Vicomtech, Spain. Since 2008, his research focuses on distributed computing and Web technologies.

Stefano Masneri received the M.Sc. degree in telecommunications from the Università degli Studi, Brescia, Italy, in 2008, and the Ph.D. degree from the Faculty of Informatics, University of the Basque Country in 2024. He is a Technical Manager for AI projects with NTT DATA and collaborates with the Computer Languages and Systems Department, University of the Basque Country, Spain. He is currently involved in projects studying how to apply generative artificial intelligence models to improve industrial processes.

Mikel Zorrilla studied Telecommunication Engineering with the University of Mondragon, Spain, and received the Ph.D. degree from the University of the Basque Country, Spain, in September 2016, titled Interoperable Technologies for Multi-Device Media Services. He is the Head of the Digital Media Department, Vicomtech, Spain.

Gabriel-Miro Muntean (Fellow, IEEE) is a Professor with the School of Electronic Engineering, Dublin City University (DCU) and the Co-Director of the DCU Performance Engineering Laboratory. He was awarded the Ph.D. degree by DCU in 2004. His research interests include quality-, energy- and performance-related issues of rich media content delivery. He is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING and the Multimedia Communications Area Editor of the IEEE COMMUNICATION SURVEYS AND TUTORIALS. He coordinated the EU Horizon2020 Project NEWTON and leads the DCU teams in the EU Projects TRACTION and HEAT. For more information, see gabriel.muntean@dcu.ie