CHINA COMMUNICATIONS

Low-Cost Tiled Display System Architecture with Dynamic Web-based Management

Daniel Marfil[1], Fernando Boronat[1,*], F. Javier Pastor[2], Anna Vidal[3]

[1] Communications Department, Gandia Campus, Universitat Poitecnica de Valencia, Calle Paraninfo, 1, 46730, Grao de Gandia, Valencia (Spain)
[2] Drawing Department, Gandia Campus, Universitat Poitecnica de Valencia, Calle Paraninfo, 1, 46730, Grao de Gandia, Valencia (Spain)
[3] Applied Mathematics Department, Gandia Campus, Universitat Poitecnica de Valencia, Calle Paraninfo, 1, 46730, Grao de Gandia, Valencia (Spain)
[*] The corresponding author, email: Fernando Boronat, fboronat@dcom.upv.es

Abstract: In this paper, a scalable hardware and software architecture for tiled display systems (a.k.a. videowalls), which can be implemented by using low-cost devices, together with a dynamic web-based management and configuration service are proposed. It has been designed to support both stored and live broadcast/broadband content, in mosaic or warp distributions. The displays and devices can be dynamically configured via web in different ways: the displays can create a single display of a larger size; or they can be configured in a customized way in order to playout different media contents in different display combinations. As display renderers, low-cost devices are proposed as the main hardware element to obtain affordable videowall systems. As a proof of concept, two prototypes have been implemented, including an accurate synchronization mechanism based on a Master/Slave control scheme and aggressive and smooth playout adjustment techniques. To evidence the good performance of the prototypes and configuration service, both objective and subjective evaluations have been conducted regarding synchronization accuracy and usability. On the one hand, the mean values of the asynchronies between the video playout processes in each display are kept below 25ms (i.e., frame accuracy). On the other hand, the obtained usability score in the System Usability Scale (SUS) test has been 88.65, which is considered as excellent.

Keywords: AMP; media synchronization; multiscreen; playout adjustment; tiled displays; videowall

Received: ?????
Revised: ?????
Editor: ????

I. INTRODUCTION

Tiled displays (a.k.a. Videowalls) consist of multiple displays that can play media content in a synchronized way, acting as one single big display. These systems are extensively used in many different sectors and for many different purposes. Among their uses, the most significant ones are the implementation of very large displays for command-and-control purposes, scientific visualization (e.g., [1–3]), collaborative and tele-immersion scenarios [4], educational purposes, advertising or commercial panels (e.g., advertising a new brand or product), touristic or informative panels (e.g., showing flight information at an airport), etc. Videowall systems can be less expensive than others, such as the ones based on video projectors [5]. For instance, they occupy less physical space and, moreover, color correction or the alignment of the involved displays are easier than when using several video projectors.

In videowall systems, the displays (i.e., tiles) can be distributed in two different ways (Figure 1). In *mosaic* distributions, displays can be uniformly distributed, as a flat or curved panel with a mosaic or matrix of MxN displays (M rows and N columns) of the same size.

In *warp* distributions, displays of the same or different size can be placed in different orientations. In addition, the displays can be combined in different ways: 1) all the displays creating a single display of a larger size; or 2) displays can be configured in a customized way in order to playout different media contents in different display combinations. Each group or combination of displays playing one single content is referred to as a *section*. As an example, in the top image of Figure 1, a 3x2 videowall with displays setup in a mosaic distribution is shown, in which two sections have been defined (2x2 and 2x1). In the bottom image, one single content is played in a videowall system with a simple warp distribution.

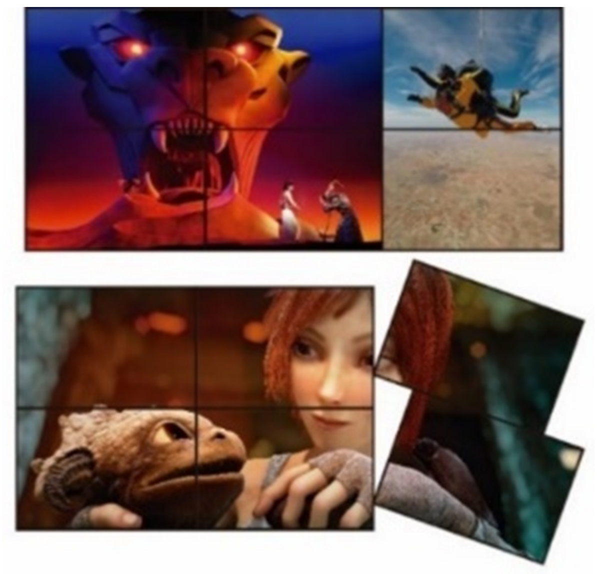


Figure 1. *2-section mosaic (top) and 1-section warp (bottom) examples of videowall display distributions*

Although there is a wide variety of commercial videowall systems, they are still not affordable for everyone due to their high cost, and, therefore, limited budget companies or individuals may not be able to acquire them. Moreover, many of them require specific hardware and proprietary software and are also based on videowall controllers (specialized or proprietary hardware) or dedicated servers. These features are the main reasons why they are expensive. Examples are Userful [1], Samsung's MagicInfo [2] or LG [3] systems. However, these solutions present some limitations that must be considered, such as: the aforementioned high prices; the fact of only being able to be locally controlled and managed (i.e., physically close to the system); the difficulty for end users, who are very likely to be non-expert, to customize or extend these systems, for example, by adding more displays; or the perception, in some cases, of the videowall as a single display by the control software, preventing the system to play out different media contents simultaneously on different sections of the videowall. These limitations and the cost-effectiveness of multi-display environments are summarized in [6]. Some of the associated challenges to provide a successful experience with this type of system are also emphasized in that work, such as scalability, ease of configuration and seamless integration of the tiled display into the environment.

In this paper, the architecture design, including both hardware and software components, of a more affordable and effective solution to implement a scalable tiled display system, allowing both mosaic or warp display distributions and different sections, is presented. It can be locally or remotely customized and managed through a user-friendly embedded web service, thus providing multi-platform access to its configuration and management service. Unlike other systems that need proprietary platform-specific applications, in our system only an Internet browser is needed. Moreover, the proposed system supports the playout of both stored and live (broadcast or broadband) content independently of its source.

When designing a videowall system, one of the main challenges is to guarantee that each portion of the video content is played out on each individual display in accurate synchronization (abbreviated as sync, hereafter) with the portions of video played in the rest of displays. Differences of few frames can be noticeable and even annoying. In the proposed system, the playout of each display is controlled by an independent low-cost electronic device, which runs an independent media player. Each device plays the content using a local and independent clock, and, although they can have the same nominal frequency, these local clocks generally present drifts, skews and imperfections, providing different clock values [7]. Consequently, this may lead to out-of-sync situations. With the aim of achieving accurate sync of the playout processes of all the involved devices, a frame accurate sync mechanism should be designed and adopted.

As a proof of concept, and in order to be able to objectively and subjectively assess the proposed architecture, two prototypes have been developed based on the use of low-cost devices and including an own sync solution. On the one hand, an objective evaluation has been conducted to check the correct playout in each display, by estimating the achieved sync level between them. On the other hand, a subjective evaluation has been conducted by users with technical and non-technical profiles. Its main aim is to obtain feedback regarding the usability of the web service, the perceived level of sync between displays and the overall performance of the system.

This paper is an extension of the conference short paper in [8] (in Spanish). The preliminary architecture and the prototype described in that paper (referred to as "the previous prototype", hereafter) have been considerably enhanced and extended with new modules and functionalities (a more complete communication protocol between entities, a more accurate and stable sync mechanism, support for live broadcast and broadband content, playlists definition, playout time scheduling and management). An own sync mechanism has been implemented (in the previous prototype, a Python-based sync library from a third-party solution was adopted) and aggressive and smooth playout adjustments (a.k.a. Adaptive Media Playout - AMP- [9]) have been adopted. In [8], only aggressive skips and pauses in the playout process were forced as playout adjustments and no subjective assessment was conducted. Additionally, in this paper, the results of the performance objective evaluation in a prototype with up to 16 displays and up to 8K video content are also included, as well as the results of the mentioned subjective evaluation. The main contributions of this paper are the following:

- Qualitative comparison between previous proposals found in related works and the one proposed in this paper.
- Hardware and software architecture of an affordable (by using low-cost devices) and scalable videowall system, allowing both mosaic and warp distributions, multiple video sections and stored and hybrid (broadcast and broadband) live content.
- An embedded web-based service for dynamic configuration of the videowall, including tools for playlists definition, time scheduling and management.
- Communication protocol and message exchange definition between the control entity and the involved rendering devices.
- The implementation of two prototypes, one 4x4 system, used for objective evaluation with 8K content, and another (transportable) 2x3 system, used for subjective evaluation.
- An own sync mechanism for both stored and live content, providing accurate and stable sync and smooth media playout in each rendering device.
- Objective and subjective assessment of the system with satisfactory results.

Unlike other works, as far as authors know, this is the first paper providing QoE related results and a subjective assessment for this type of systems.

The content of the paper is organized as follows. Previous related works are presented in Section II. In Section III, the proposed architecture design of the videowall system is described. The embedded web service for the control and management of the videowall is summarized in Section IV. The implemented prototypes, the sync mechanism included, the methodology of assessment and the obtained results are presented in Section V. The paper ends in Section VI, drawing some conclusions and pointing out some future work.

II. STATE OF THE ART

This section includes a review of different available architectures and previous works regarding videowall systems, together with a qualitative comparison of their advantages and drawbacks, including the one presented in this paper. Works describing middleware to implement tiled-video systems (such as the ones in [10–15]) have also been consulted but the inclusion of their description in this paper was discarded because the design of another middleware is out of the scope of this paper. Nevertheless, due to the similar functionalities that are provided by the SAGE2 system (described in [11]), it was tested by the authors. When high-performance devices (e.g., laptops or desktop computers) are used as display renderers, that system worked fine. Unfortunately, low-cost devices do not have enough resources to playout media content smoothly when running that system.

2.1 Videowall System Architectures

Currently, two main different options regarding media delivery and content distribution technologies for videowall systems can be found[4], such as:

1. Hardware-based solutions. Media content distribution follows hardware video standards (i.e., based on the use of video splitters or specific interconnected displays).
2. Non hardware-based solutions. Two possibilities can be found in this option: IP-based solutions, in which media content is delivered via IP streaming; and stored content-based solutions, in which media contents are available in (remote or local) media repositories or media units.

2.1.1 Hardware-based solutions

Regarding this option, on the one hand, splitters can be used. Splitters are devices which usually have one input and several outputs (each one connected to one display). The input receives the original media (video) content in full resolution (i.e., without resolution or size modifications and without any cropping). Then, the splitter's main function is to crop the video frames and send (through its outputs) the corresponding part of the video to each display. This option is designed for simple configurations in which all the involved displays are similar. It has several drawbacks to be considered: 1) lack of scalability, as the splitter has a finite number of outputs; 2) specific hardware is required (a splitter); 3) the source of the content must be close to the videowall system; and 4) the configuration options are quite limited, e.g., if more than one video section is considered for the videowall, the use a multiplexor is needed (i.e., drawback #2).

On the other hand, several displays can be interconnected, usually through a proprietary communication bus, in order to create a videowall system. This concept is used in proprietary solutions that some manufacturers have implemented (e.g., LG or NEC). For instance, NEC's solution allows the implementation of 10x10 videowall systems by interconnecting 100 displays in series. These systems are very expensive, and the involved displays must be from the same manufacturer. Moreover, there is an issue associated to their maintenance. As probable faults or damages may happen to the videowall system, it can imply replacing the displays with very similar ones (or even requiring exactly the same model).

2.1.2 Non hardware-based solutions

IPstreaming-based solutions These solutions are based on the distribution of media content via IP streaming, are scalable and adaptive. Samsung's MagicInfo system is an example and supports up to 250 large displays. Moreover, the installation is easy and simple. It only requires the interconnection between all the involved displays through an IP network (Local Area Network or LAN). However, that network has to provide enough bandwidth to be able to deliver the content simultaneously to the different connected displays. In order to minimize the needed bandwidth, only the corresponding part of the video to be played in each display can be sent to it. This requires one or several powerful computers (depending on the size of the videowall) that crop the media content in different streams (one for each display) and send them through the network. The need of these computers increases the final cost of the system. Besides, in most commercial systems, a specific hardware device is connected to each display (e.g., devices known as Zero clients) with specific SOFTWARE capable of decoding and rendering the received video streaming. Among many others, examples of videowall systems based on this option are the aforementioned MagicInfo, Userful systems and [16].

Stored content-based solutions When only stored content is used, no powerful computers are needed and, therefore, the final cost of the videowall system is significantly lower. In this case, a device capable of accessing the content located in a shared (local or remote) media unit can be connected to each display. In such a case, either each device or another control device can select the part(s) of the video to be played, without the need of using IP streaming. This way, a videowall system can be implemented with low-cost devices, which always have limited resources. Only a LAN with access to the media content, providing enough bandwidth to allow all the devices to simultaneously download the selected content, is required.

2.1.3 Design choices for the proposed system

Regarding the three analyzed options above, the first one was discarded for the proposed system because it is not flexible nor scalable enough. Nowadays, low-cost electronic devices with high speed LAN connectivity and internetworking are already available in the market. These devices provide enough bitrate to work with stored content, as well as to upload or download content to or from a connected media unit during the videowall system's operation.

Therefore, the system proposed in this paper follows a combination of the two options of non hardware-based solutions. On the one hand, it has been designed to work in a wired LAN, where stored content can be available. On the other hand, in order to support the playout of live content, IP streaming through the LAN has also been included. This is possible thanks to the current cheap enough commercial switches, which provide the required bandwidth in the LAN environment without further issues. Moreover, an additional device is also required in order to stream (and optionally transcode) the content through the LAN.

2.2 Sync Strategies and Previous Related Work

In this Subsection, a brief description of existing sync strategies and previous related works, based on the use of low-cost devices, are presented.

2.2.1 Sync strategies

Regarding sync control strategies, there are two different types: centralized and distributed strategies [17]. On the one hand, two different centralized schemes can be adopted: the Master/Slave scheme (M/S, hereafter) and the Synchronization Maestro Scheme (SMS, hereafter). The M/S scheme involves one of the playout devices becoming the sync reference by providing its own playout timing as the reference. It should provide the remaining involved playout devices (i.e., the slaves) with the necessary information to let them make appropriate playout adjustments to achieve accurate sync. In the SMS scheme, there is an independent device (the maestro device) in charge of receiving timing information from all the playout devices, processing it and sending them a reference playout timing to make the appropriate adjustments. On the other hand, the Distributed Control Scheme (DCS, hereafter) implies that all the involved playout devices exchange their timing data. When a playout device has collected the timing information from the rest of devices, it will process it and calculate the reference playout timing to make its own playout adjustments. For the SMS and DCS schemes, the timing reference can be computed by adopting different criteria. For example, it could be the playout timing of the most lagged or advanced device, or the mean value of the timing for the involved devices. In the case of the M/S scheme, it is the playout timing of the master device. An extended explanation of these types of sync control schemes and criteria for selecting the reference can be found in [17, 18]. In [18], authors provide an exhaustive qualitative comparison between centralized and distributed sync control schemes when used for inter-destination media synchronization, considering some key factors, such as robustness, scalability, traffic overhead, flexibility, location of control nodes, interactivity, consistency, causality, fairness, coherence, and security.

2.2.2 Previous works

In [19], a videowall control system based on server virtualization is proposed. The Synchronized Multimedia Integration Language (SMIL [20]) is used to manage the metadata associated to the spatial layout of the media content, as well as for temporal sync. However, no results regarding sync performance are provided. Similarly, in [21]] a videowall system based on a virtual machine is described. In that virtual machine, the content to be played in the tiled display is hosted. The virtual machine, which runs in an Intel Core i7 CPU and 12GB of RAM computer, generates and encodes the streams. Raspberry Pi (abbreviated as RPi, hereafter) devices are connected to the involved displays. Unfortunately, sync performance results are not provided either.

In [16], a low-cost videowall system based on RPi is proposed. It adopts a M/S sync scheme between the displays and the system can also be configured via web. In that work, according to their authors, accurate sync is achieved but no rigorous measurements are provided. Unlike the system proposed in this paper, functionalities such as live content, playlist configuration, display rotation, warp distribution or the definition of different and independent video sections are not implemented nor described in that paper.

The RPi-based videowall systems presented in [22–24] support warp distribution and also implement sync mechanisms, but they do not allow display rotation or dynamic/remote web-based configuration. None of these works provide sync performance results and do not explicitly indicate which sync control scheme they use, although it can be deducted that they are adopting a M/S sync scheme. On the one hand, the systems proposed in [22, 23] do not support image slideshow, playlist generation, live content playout or video sections. In particular, in [23] the videowall configuration is described in a text file. When a configuration modification is performed, a reboot of the involved devices is required, or they are forced to re-read that text file. In that paper, although RPi devices are also adopted to play the content in each display, the adoption of a more powerful computer as the control device is recommended, resulting in an increase in the cost of the system. On the other hand, in [24] nothing about those features is said, so authors cannot make any statement about its inclusion.

In [25] a sync method is proposed for irregular screen systems (which can be directly applied to warp -or mosaic- distributed videowalls). An SMS scheme is adopted and, in order to predict the individual playout latency for every involved device, machine learning techniques are used by the maestro device (requiring high processing power). For that purpose, the playout devices must send feedback messages to the maestro device including data regarding their decoding rate (or speed). A very accurate sync level is achieved (less than 10ms, as stated by the authors in that work), however, the experimental results are carried out with up to 12 displays and computers with higher performance specifications (e.g., i3, i5 and i7 CPUs) and more expensive than the ones that are proposed to be used in the architecture presented in this work.

A different work, presented in [26], proposes an SMS control scheme based on the use of a control server that gathers the V-Sync generating time of all the playout devices and predicts the timing of the vertical sync signals of each involved rendering device. Authors in that work claim to achieve a sync level of less than 1 frame.

Regarding other types of sync mechanisms, authors in this work know the sync approaches presented in [27–29], nonetheless, their discussion has not been included in this section, as their scope (i.e. second screen experiences while consuming TV content) is not the same as the one in this work.

To sum up, Table I presents a qualitative comparison between the related works described in this Section and the videowall system proposed in this paper. The adopted criteria to specify the comparison have been some features that authors consider as relevant, since authors have not found any previous work including a similar explicit comparison or defining any standardized criteria, specifications or functionalities used for making such a comparison. The selected features are the ability to divide the videowall into video sections; the support of streaming technologies; the inclusion and explanation of the proposed sync mechanism; warp distribution support; the generation of playlists; the support to dynamically change the videowall number of involved displays; the web-based configuration or the inclusion of a prototype. A tick (V) means that a specific functionality is implemented and a cross (X) means that the work does not include the functionality or that there is not enough information in the corresponding paper to assert its implementation.

III. PROPOSED DESIGN

In this section, the functionalities and the architecture design (including both hardware and SOFTWARE components) of the proposed system are presented.

3.1 Functionalities

In the proposed scalable system, any possible combination of the displays can be selected: a warp distribution, a mosaic distribution or a combination of both; either grouping some of the displays together to playout different media contents (i.e., with different videowall sections) or grouping all together playing out the same video, becoming a seamless larger display, as shown in (Figure 1).

Through an embedded web service, the videowall system can be locally or remotely managed and customized, by defining one or several sections and selecting the content (TV, video, picture, etc.) to be played out in each of them, whatever the videowall configuration is. This management service also allows the programming or time scheduling of independent

Table 1. *Related Work Functionalities comparison*

Compared Work	Multiple Video Sections	IP Streaming	Sync	Warp Distribution	Playlist &Content Scheduling	Dynamic Configuration Configuration	Web-based Configuratio
i2CAT [16]	X	V	V	X	X	V	V
Kwang-Yong et al. [19]	X	V	V	X	X	V	X
Bundulis et al. [21]	X	V	X	X	X	V	X
Pi Wall [22]	X	X	V	V	X	X	X
Alabdulsalam et al. [23]	X	V	V	V	V	X	X
Mihalopoulos [24]	X	X	V	V	X	X	X
Shin et al. [25]	X	V	V	V	X	V	X
J. Kanda et al. [26]	X	X	V	X	X	X	X
Proposed System	**V**	**V**	**V**	**V**	**V**	**V**	**V**

playout sessions in each defined section of the videowall. Video or image playlists can be created and selected to configure a set of media content to be played out consecutively. Additionally, media content can also be configured to be played in loop or at a specific time. Additionally, the source of audio to be connected to the videowall speakers can also be selected (explained in the next Subsection).

Regarding the available contents, as previously mentioned, they can be stored or live (i.e., broadcasted TV or broadband content). Stored contend can be locally available in local media units (e.g., a hard drive disk or HDD) or they can be remotely available (e.g., in the cloud). In both cases, media contents must be reachable by every low-cost display device involved in the system. In order to support live content, an IP streaming service is included.

Furthermore, the system has been designed to enable scalable videowall configurations (e.g., useful when a user wants to enlarge the videowall). For instance, if a new row or column of displays are added to the videowall, this can be easily managed by using the web service. The end-user is not required to be an expert, as the only steps to follow are: 1) copy the HDD disk (or SD card, when RPis are used) image from an already configured low-cost rendering device to the disk (or SD card, when RPis are used) of the new ones (one per additional display); 2) setup the IP configuration correctly for each new device, and connect them to the corresponding display and to the videowall LAN; and 3) configure the videowall system accordingly, through the web service by using any Internet browser (explained in Section IV).

3.2 Architecture Design

The architecture proposed for the videowall consists of two parts: hardware and software (see (Figure 2)), each with its elements or modules. In this section, both parts are described.

3.2.1 Hardware part of the architecture

In order to manage and control the playout on the displays involved in the videowall system, the use of low-cost commercially available hardware devices and a private high-speed LAN scenario are proposed. In total, there are as many low-cost devices (named *render devices*) as displays. The video output of each *render device* is connected to the video input of each display. Moreover, an additional device has been included, called *Videowall Content Provider Unit* (abbreviated as *VCPU*, hereafter). *Render devices* and the *VCPU* are interconnected through an IP network.

(Figure 2) shows an overview of the hardware components for a generic MxN videowall system in a mosaic distribution. In that figure, the system hardware consists of MxN displays, each with its *render device*, the *VCPU*, an IP network and an external user-device, such as a smartphone, laptop or PC, with a browser and IP-based connectivity to the *VCPU*. In the following lines some of the main involved hardware devices are described.

The maximum resolution of the video that can be played in the overall videowall system will depend on the maximum resolution of the portion that each *render device* is able to render and the number of *render devices* in each row and column.

Videowall Content Provider Unit (VCPU) The *VCPU* hosts the web-based management service and provides the media content to the system through the videowall's private LAN. It can be connected, if required, to any local or external storage unit via USB or via IP. If the content is stored anywhere (e.g., in the cloud), it will download it in advance and store it to make it reachable by the *render devices* through the videowall's wired LAN.

The *VCPU* also receives the TV broadcasted content and streams it to the *render devices*, when live TV content is selected. To receive it, a DVB-T/S receiver (e.g., USB or PCI card) will be used.

In order to allow remote configuration and management of the videowall, when needed, the *VCPU* can be connected to a Wi-fi Access Point (local access) or to the Internet through another public interface/network (worldwide access).

Display Render Device Every *render device* is connected to its corresponding display (e.g., via HDMI) and to the videowall's LAN (e.g., through a switch). Regarding the IP interconnection, all the involved devices can be set up with static IP addresses in an IP private network (e.g., the 192.168.0.0/16 network).

On the one hand, in order to easily identify the involved *render devices* in a videowall system with a mosaic distribution, the IP address of each *render device* can be configured, as e.g., 192.168.X.Y, where, for simplicity, X can be the row number and Y can be the column number (starting from the top left corner of the videowall). This type of configuration makes the management of the videowall system easier, although random IP addresses can also be assigned since the number of each *render device* and the assigned IP address are stored in a *configuration database* in order to identify them (see Section IV). Taking advantage of the use of this database, in videowall systems with warp distributions, the IP address assignment can follow any logic.

Electronic Circuit for Audio Output Selection In order to manage and select the audio output of the system speakers, a simple electronic circuit has been designed. This circuit is known as a *summing amplifier circuit* (shown in Figure 2), where the output voltage is proportional to the sum of the input voltages. In particular, two independent circuits are needed for each audio channel (Left and Right). As an example, an MxN videowall would require MxN audio outputs. By using the summing amplifier circuit, the output of each *render device* is connected to the corresponding (left and right) audio input channel of the circuit. Each circuit has (MxN)+1 impedances, whose value (r) is not significant as long as they are all the same in the whole circuit. This implies that no amplification nor attenuation will affect the audio signals. Through the management web service, the audio output of the speakers of the videowall can be selected, in order to choose which of the sound outputs will be unmuted (the rest will remain muted).

Moreover, multiplexors could also be used in order to have more than one group of speakers available, so many audio outputs could be redirected to a different specific group of speakers (or audio interfaces, such as Bluetooth or radio output channels). This enhancement has been left for further work. It can be interesting, for instance, in videowalls located in public venues (e.g., airports) in which several videos are being played out. This way, a user could choose the audio track of the specific video she/he is mainly watching by, e.g., tuning the appropriate radio channel in its own device.

Use of low-cost devices The cost of the hardware of the videowall mainly includes the costs of the screens (the most important part, depending on their size and features), the cost of the speakers (including the electronic circuit for audio selection), the cost of the infrastructure to attach the screens in the desired distribution (mosaic, warp or hybrid), the cost of one render device per screen, the cost of the VCPU, a Digital TV receiver card, and the cost of the networking equipment.

By using low-cost devices, a more affordable videowall system can be obtained. For example, RPis can be used as render devices. The need of processing resources and memory requirements in the VCPU is not significantly affected by the size of the videowall (i.e., the number of tiles). Its main function is to configure and control the videowall through a web service, regardless its size. On the one hand, for stored content,

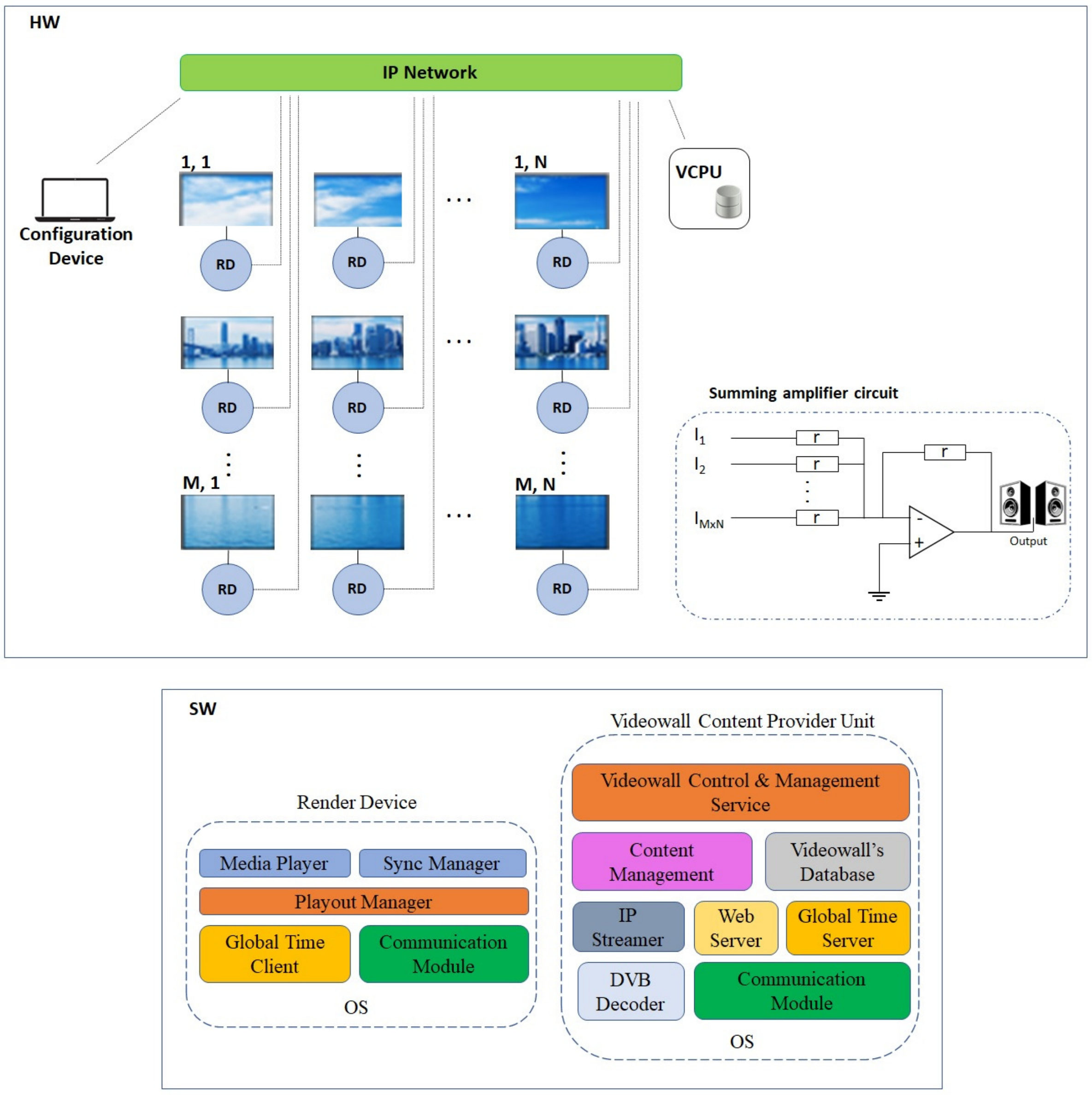


Figure 2. *General overview of the proposed hardware architecture (top) and the involved software modules (bottom)*

it only provides the render devices with the network path to the content (e.g., in a local media server or in the VCPU itself), which can be prepared or processed (e.g., to be rotated or divided in tiles, if needed) in advance. On the other hand, for live content, it receives, transcodes and streams it to a multicast address, independently to the number of tiles in the section it is sent to. A low-cost mini-PC can be used as VCPU. For receiving broadcast TV content, a DVB-T receiver is needed (e.g., a cheap DTV USB receiver) [5]. Regarding the networking equipment, only LAN equipment would be required. A simple 100Base-T switch with a number of ports higher than the number of devices to be networked could suffice. If more bitrate is needed for some device (e.g., for a media server, if included) a switch with 1000Base-T ports could be used. For wireless access (if needed, for wireless management), just a cheap WiFi Access Point would be needed. Finally, in case remote configuration and control of the videowall is needed, the VCPU would need an external connection (e.g., to the Internet). Notice that, as time goes by, these needed components are getting cheaper.

In Subsubsection 5.1.1 the description of the cheap hardware components selected for the implemented prototypes is provided.

3.2.2 Software part of the architecture

In this sub-section, the involved software components are explained. Figure 2 bottom shows the involved software modules needed to implement the proposed videowall system, in both *render devices* and *VCPU*. In the following lines, all these modules are explained but the *Videowall Control Service Module*, which, due to its relevance, is explained in more detail in Section IV. The modules are described in this order: first, common modules for *render devices* and *VCPU*; then, the modules which are only set in the *VCPU*; and finally, the modules which only involve the *render devices*.

Operating System (OS) On the one hand, the OS running in the *render devices* is very important and should be analyzed when implementing the proposed system. It will determine the kind of software components they can run. Most of the low-cost devices currently on the market can run Linux-based OSs. For example, if RPi devices are used, a Debian-based OS called Raspbian [6] can be installed. On the other hand, the *VCPU* is a computer and could run any OS for PC (e.g., Linux, MS Windows. . .).

Communication Module This module also involves both *render device* and *VCPU*. In the *VCPU*, the web configuration service is hosted. A communication channel will be established between each *render device* and the *VCPU* (between their *Communication Modules*).

When a *render device* boots, it will remain waiting for a message from the *VCPU*, depending on the videowall system configuration or setup. Through the established communication channel (i.e., in a unicast way), three different types of text messages, starting with “VIDEO”, “AUDIO” or “IMAGE” words, can be received, depending on the nature of the content to be played out. Following these words, some parameters are provided in each message, delimited by the ‘%’ character. When receiving VIDEO or IMAGE messages, if the involved *render devices* are playing a different content, they will stop that playout and start playing the new content at the indicated instant.

• Video-related messages: The format of the video related messages is the following:

”VIDEO%uri%X1%Y1%X2%Y2%position %multicast_ip:port% time%master”

The *uri* parameter contains the Universal Resource Identifier (URI) of local or remote content, or a live stream’s IP multicast address (e.g., *rtp://239.1.1.9:5004*). *X1* and *Y1* parameters represent coordinates indicating the top left corner of the video content to be played, while *X2* and *Y2* parameters represent the coordinates of the bottom right corner of the video content to be played. The *position* parameter indicates where the content should be played (L, R, T, B or C values, for left, right, top, bottom, or center, respectively). It allows the adjustment of the content portion’s position in the displays for specific cases. Examples of these cases are peripheral displays of the videowall system or in warp displays distribution videowalls, where the displayed content may not occupy 100% of the display (i.e., full screen). The *multicast_ip:port* parameter includes the multicast IP address and port number to be used by the sync messages exchange process (explained in Section IV).

The *time* parameter contains the NTP-formatted instant (according to the global clock reference) when the playout should be started. This latter parameter has been included to force all the involved displays to start their playout at exactly the same time (according to their global time clock reference). The *master* parameter is a boolean variable to indicate whether a particular *render device* holds the *master* role or not (this is explained in Section IV in more detail). For example, for a mosaic 2x3 videowall system composed of 6 displays with an aspect ratio of 16:9, the message that the top right display (with the 192.168.1.3 IP address) will receive, if the whole videowall plays a 1920x1080 resolution video, could be:

"VIDEO%http://192.168.0.2/media/video1.mp4 %1280%0%1920%960%L%239.0.0.15:5004 %3701255471291907022%false"

• Audio-related messages: The format of the audio-related messages is the following:

"AUDIO%action"

The action parameter can contain values such as mute, unmute or volume:value, in order to switch off/on the audio of each render device or to change the volume, respectively.

• Image-related messages: The format of the image-related messages is the following:

"IMAGE%uri%X1%Y1%X2%Y2%position%time"

The *uri* parameter will contain the URI where the picture is (locally or remotely) located, and *X1*, *Y1*, *X2*, *Y2*, *position* and *time* will have the same meaning as in the video-related messages. In this case, as the content to be displayed is a static image in a videowall section (or maybe in the whole videowall system), there is no need to implement sync mechanisms. It will only be necessary to sync the instant (according to the global clock reference) when the image will be displayed.

Global Time Modules To get accurate inter-device sync, all the playout processes in each *render device* should follow a common global clock reference (provided by a local Clock Server as, e.g. a local NTP server), and an appropriate inter-device sync mechanism.

In the proposed system, every r*render device* should have its internal clock synchronized through a *Global Time Client* module connected to a *Global Time Server* module located in the *VCPU* (e.g., a Network Time Protocol or NTP server [30] or Precision Time Protocol or PTP server [31]). In the presented low-cost design, the use of NTP is proposed, since the accuracy that can be achieved in LAN environments by using a local NTP server meets the videowall system requirements. Moreover, it does not require any specific hardware, unlike in PTP, which would increase the cost of the system.

Content Management Module This module, as seen in Figure 2 bottom, is only set in the *VCPU*. To allow the use of stored contents, the videowall system can include a local or remote media unit. Then, an *FTP server module* can be enabled (e.g., in the *VCPU* if local disc is used) to manage the stored contents. Different permissions can also be set for each authorized user, and, depending on the permission level, each user will access only to some specific folders. This way, authorized users will only access to their associated media content when configuring the content to be displayed in the different sections of the videowall.

Additionally, if there is a content stored in a remote resource (i.e., in the cloud), when it is selected to be played out in a section, it has to be previously downloaded and stored. For this, a *media downloader element* has also been included in this module.

Modules for Live Content On the one hand, in order to receive and prepare the live TV broadcasted content, a *DVB-T/S decoder module* should be included in the *VCPU*. Additionally, an *IP streamer module* is needed to stream broadcast TV content (and transcode it, if necessary) to the *render devices* (e.g., via multicast RTP [32]) through the LAN. On the other hand, the players in the *render devices* are also required to be able to receive and decode streamed (RTP) content. Regarding live content (not only broadcast, but also broadband content), *render devices* are agnostic from the source or delivery technology providing the content, as long as the *VCPU* is capable of converting it to IP multicast RTP-based streams.

Media Player In order to select the most suitable media player to be run in each *render device*, players including inter-device sync mechanisms or players that can be extended to include them, when devices are connected to an IP network, should be considered.

Playout Manager This module is in charge of launching the media player and managing the media content to be played in it, according to the messages received from the *VCPU*.

Sync Manager In order to achieve accurate sync, one of the aforementioned sync control schemes (i.e., M/S, SMS or DCS) and strategies for selecting the timing reference and playout adjustment techniques should be adopted. If the M/S scheme is adopted, one of the *render devices* of each videowall section will behave as the *master* device, send its own playout timing information to the other *render devices* that will act as slaves. The slave *render devices* will adjust their playout process to achieve sync with the one of the master *render device*. Similarly, if the SMS scheme is adopted, the sync reference should be computed by an external entity (e.g., the *VCPU*) from data gathered from all the *render devices*. That entity will send the reference to all of them to force the appropriate playout adjustments. Finally, if the DCS scheme is adopted, the sync reference will be computed by each *render device* from data gathered from all the other *render devices*. Independently of the adopted sync scheme, the sync control information should be exchanged between the devices in each section via multicast to be bandwidth effective.

If any asynchrony situation is detected by the sync manager module of a *render device*, it will adjust its video playout process. An asynchrony situation is detected when the playout process of *a render device* is advanced or lagged regarding the playout process of the received or calculated sync reference in the same section, exceeding a configured asynchrony threshold. Every time an asynchrony situation is detected and processed, this module will send the media player the corresponding type and magnitude of the playout adjustment to be carried out in order to correct it. Two types of adjustment to achieve sync can be performed: aggressive (skips and pauses, -S&P-) or smooth (Adaptive Media Playout, AMP [9]) adjustments. On the one hand, S&P adjustments consist of skips or pauses in the playout processes, when the playout timing reference (i.e., *master render device* playout timing) is advanced or lagged exceeding the configured threshold, respectively. These adjustments are more noticeable and may result annoying for users. On the other hand, AMP adjustments consist of modifying the playout rate by increasing or decreasing it during a small interval of time, in order to reach the playout timing reference. This type of adjustments is less noticeable and, therefore, less annoying for users [9].

The selection or comparison of different schemes for sync control or techniques for sync reference selection or for playout adjustments in these types of systems is out of the scope of this paper. In order to validate the proposed architecture, in the implemented prototype presented in Section V, a simple M/S sync control scheme and both types of playout adjustments have been included. It has been chosen because, according to the work presented in [18], that scheme can provide the best performance in terms of scalability, traffic overhead and interactivity (low delays) when used for inter-destination media synchronization purposes.

Database Module In order to store all the registered users' relevant information, a *user database* is included. Parameters such as the login credentials or the user's profile are stored in it. For instance, a SQL-based Server can be set into the *VCPU* to store this information, which will be consulted to enable (or disable) the different implemented functionalities, depending on the user's permissions.

Additionally, to save the different possible configurations (e.g., playlists, time scheduling, etc.) for a determined videowall system, an additional database is also included to store this information (*configuration database*).

Web Server The *Web Server* module hosts the web-based control and management service module.

Control and Management Service Module Due to the importance of this module, it is explained in detail in the following Section. This module and the previous one (Web Server) contribute to meet two of the initial requirements for the proposed system that were: 1) to provide a multi-platform/device access to the configuration of the complete system and 2) to enable remote configuration.

IV. WEB-BASED MANAGEMENT SERVICE

In this section, the developed web-based Videowall Control Service Module (abbreviated as VCSM, hereafter) is explained. The used technology to implement the VCSM is HTML5. This way, end-users do not need to install any third-party dependency or a specific application but a web browser.

A user access control mechanism has been included, so different users with different roles can access the videowall system with different configuration rights. An *Administrator* user (with *Admin* profile) has been set in order to manage the whole videowall system, as well as being able to add, modify or delete any other user and the associated profiles. Some ordinary users (i.e. users without full administration rights) will have permission to access and play a particular content while other users will not be able to even access the path where that content is stored.

Once the credentials have been provided through a login screen, another screen with up to four menu options will appear, depending on the user's role. This screen includes videowall distribution and displays configuration options (only for the users with *Admin* profile), such as management options and, overall videowall configuration. Moreover, this screen also includes another option available for all users to select content.

4.1 User With Admin Profile

The videowall system can be configured in either mosaic or warp distribution. Figure 3a and Figure 3b show the general configuration screen of the videowall system with both distributions. Only the users with *Admin* profile can access to this screen. The sync parameters (explained later in Section V) can also be configured, i.e., the *master render device* transmission interval of its playout timing information, the asynchrony threshold value and the guard interval period.

Regarding the mosaic distribution (MxN), the number of horizontal (M) and vertical (N) displays can be selected. Regarding videowall systems with warp distribution, the displays can be added one by one by just indicating their size and number. They can be dragged and dropped in a panel, as well as setting a rotation angle for each one in order to set them as they physically are in the videowall system. This way, displays can be independently selected, and their individual properties can be modified (i.e., size, position or rotation).

Moreover, users with an *Admin* profile can access to a configuration menu where the user's properties are managed (Figure 3c). The *users with Admin profile* can add, modify or delete other users of the system. Moreover, the user's profile is also set.

4.2 Ordinary Users

Users with an ordinary role can only access to a menu where the video sections, the configuration of playout schedules or playlists and the available content for them can be selected. Figure 4 top shows the screen with the videowall system distribution once it has been previously configured by a user with Admin profile (in this example, a 2x3 mosaic distribution), where any of the involved displays in the videowall sections can be selected. Additionally, Figure 4 bottom shows different section configurations for a 2x3 mosaic videowall: 1) shows a 2x3 section; 2) shows a 2x2 and a 2x1 sections; 3) shows two 1x2 and a 2x1 sections and 4) shows three 2x1 sections.

Sections in a mosaic distribution can be easily created selecting the top left and bottom right displays delimiting the section (in this order) by left clicking on it with the mouse. Then, stored or live content can be selected to be played out in that section (buttons "Select stored content" or "Select live content", respectively). A pop-up screen will show the available contents to be selected. Additionally, the end user can select which *render device* to listen to (button "Select audio"), and the rest of the *render devices* will remain muted. The audio of the *render device* with the lower IP address (usually the one in the left top corner, in a mosaic videowall or the first one added in a warp distribution) is selected by default.

In case the videowall system follows a warp distribution, the involved displays in each videowall section must be selected one by one. Then, the video section is specified, and the selection of the media content can be configured similarly to the mosaic distribution configuration steps.

In this screen, there are also control buttons regarding the videowall system playout (i.e., Play and Stop buttons), as well as three additional options: "Schedule configuration", "Load Configuration" and "Reset":

- *Schedule configuration*: allows the videowall system to be set automatically by creating a playlist. This way, different configurations can be temporally presented in the videowall by playing out media content consecutively. That is, the videowall system can be configured in different sections with different media content, that will be shown automatically and continuously since the configured date and time for the first media content in the playlist. Besides, in this section independent media content can also be configured to be played in loop (this can be understood as a particular type of playlist). To create a playlist (or schedule media content in loop), a new event (i.e., scheduled content) can be added in this configuration menu. Additionally, date and time information, and the corresponding video section is also selected. This information is stored in the *configuration database*.
- *Load configuration*: this option lets the user select and load one of the stored configurations in the system. Once a configuration is loaded, the *VCPU* controls the changes in content's playout in each section of the videowall, depending on the time of the day.
- *Reset*: This option allows users to reset the videowall system's playout and the loaded configuration. It stops all the current playout processes and clears the videowall configuration. This functionality can be used, e.g., when a new display is added to the videowall system.

V. EVALUATION

In this section, the implemented RPi-based prototypes, the included own sync mechanism and its configuration parameters, the followed evaluation methodology and the obtained results of both objective and subjective evaluations are presented.

5.1 Implemented Prototypes

In order to validate the proposed videowall system, two different prototypes have been implemented, one for the objective evaluation and another one for the subjective evaluation process. On the one hand, a 4x4 mosaic videowall system (using all the RPi 3 [7] devices available at our lab) has been implemented for the objective assessment process, in order to evaluate the performance of the proposed system in a quite large configuration. As the goal was to assess the achieved sync level, displays have not been fixed to any structure and 16 displays in a lab room in the author's University Campus have been used. On the other hand, for the subjective evaluation process a portable 2x3 mosaic videowall system has been implemented. The main reason for choosing such a small system is that it should be moved to be assessed in different locations of interest in the Campus. At the research group, only a structure for a small 2x3 videowall was available and it was adapted to a platform with wheels. Figure 5 shows the two implemented systems. Later, in Subsection 5.3, more pictures of the portable 2x3 videowall are provided.

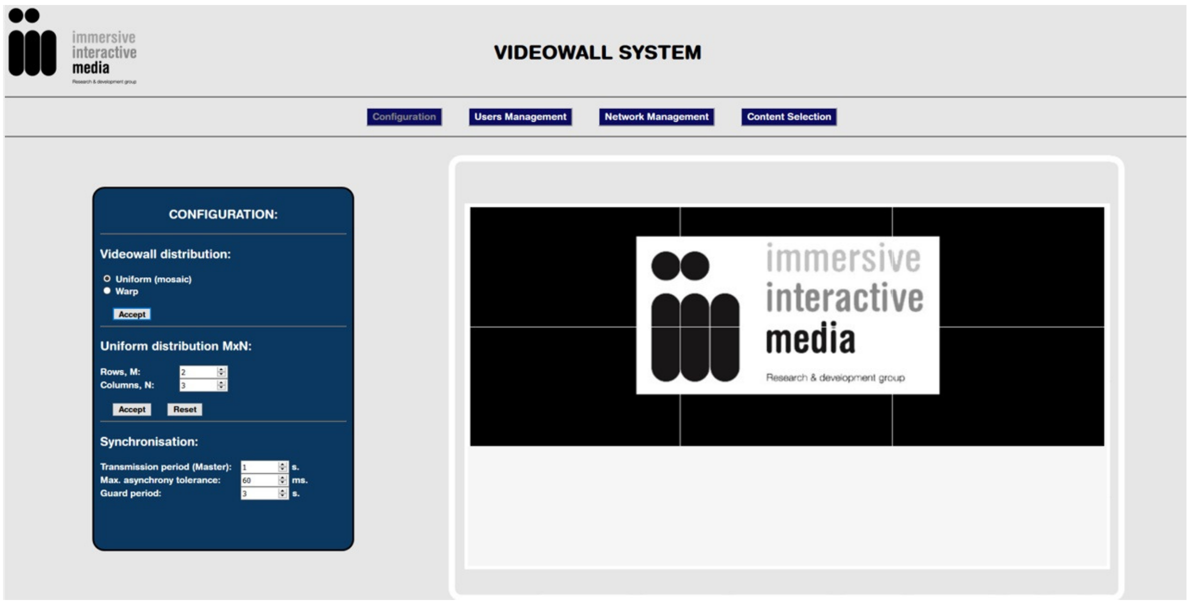
5.1.1 Involved hardware

Up to sixteen displays have been used in order to implement the two aforementioned videowall systems. Therefore, sixteen *render devices* have been configured and connected, in addition to a *VCPU*, a LAN switch and a laptop as the end-user's device. Additionally, during the subjective evaluation, participants have used their own desktop computers or personal devices. As low-cost *render device* and *VCPU* devices, 16 RPi Model 3B and an Intel NUC device (with an N3050 processor, 4GB RAM and Ubuntu 14.04 OS) have been chosen, respectively. The WinTV Nova-T USB dongle has been used as the DVB-T receiver. The involved displays are 16x Samsung S22D300Hy (22") and 6x LG M2432 (24"). Moreover, for connectivity purposes, a 100Mbps D-Link switch and a 300Mbps TP-Link wireless router have been used.

5.1.2 Involved software

In this sub-section the involved software for the implemented prototype is explained.

Media Players Regarding the media players, two different tools have been included: one to play video contents and another one to view images. Regarding



CONFIGURATION:

Videowall distribution:

[✓ Uniform (mosaic)]
Warp

Accept

Uniform distribution MxN:

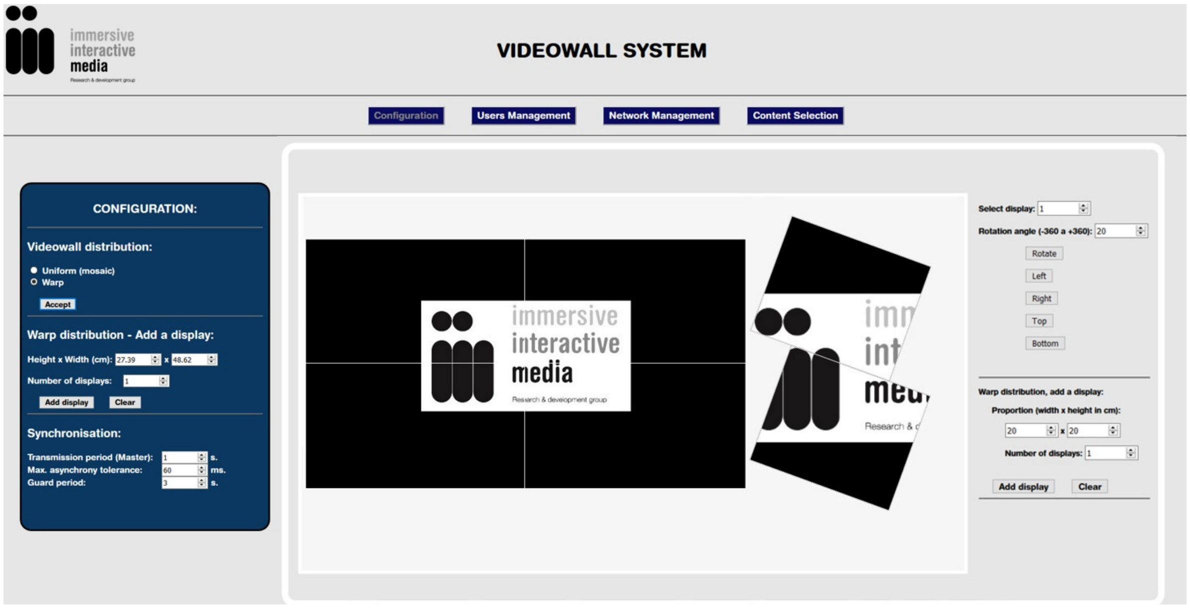
Rows, M:	2
Columns, N:	3

Accept Reset

Synchronisation:

Transmission period (Master):	1	s.
Max. asynchrony tolerance:	60	ms.
Guard period:	3	s.

(a)



CONFIGURATION:

Videowall distribution:

Uniform (mosaic)
[✓ Warp]

Accept

Warp distribution - Add a display:

Height x Width (cm): 27.39 x 48.62

Number of displays: 1

Add display Clear

Synchronisation:

Transmission period (Master):	1	s.
Max. asynchrony tolerance:	60	ms.
Guard period:	3	s.

Select display: 1

Rotation angle (-360 a +360): 20

Rotate
Left
Right
Top
Bottom

Warp distribution, add a display:

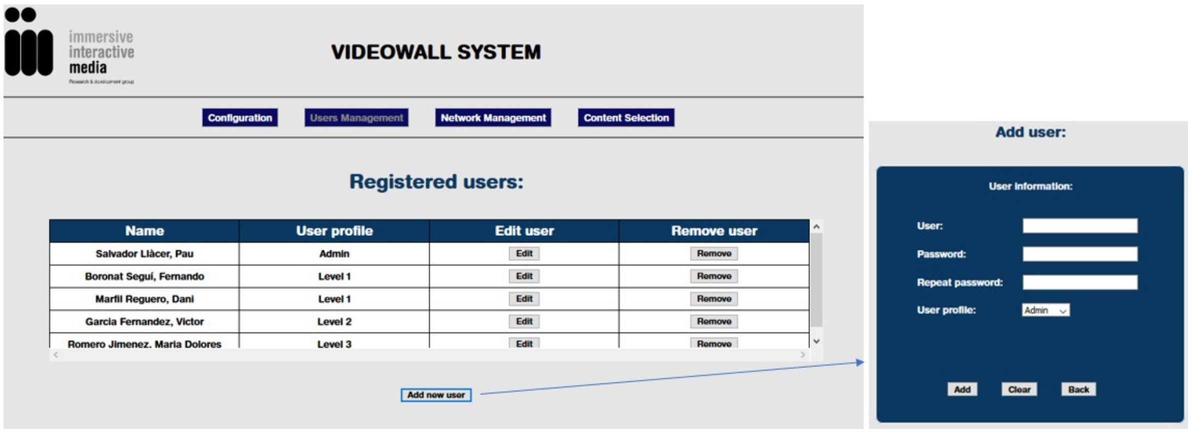
Proportion (width x height in cm):

20 x 20

Number of displays: 1

Add display Clear

(b)



Registered users:

Name	User profile	Edit user	Remove user
Salvador Llàcer, Pau	Admin	Edit	Remove
Boronat Seguí, Fernando	Level 1	Edit	Remove
Marfil Reguero, Dani	Level 1	Edit	Remove
Garcia Fernandez, Victor	Level 2	Edit	Remove
Romero Jimenez, Maria Dolores	Level 3	Edit	Remove

Add user:

User information:

User:	
Password:	
Repeat password:	
User profile:	Admin

Add Clear Back

(c)

Figure 3. *Mosaic (a), warp (b) distributed videowall configuration screen and users configuration screen (c)*

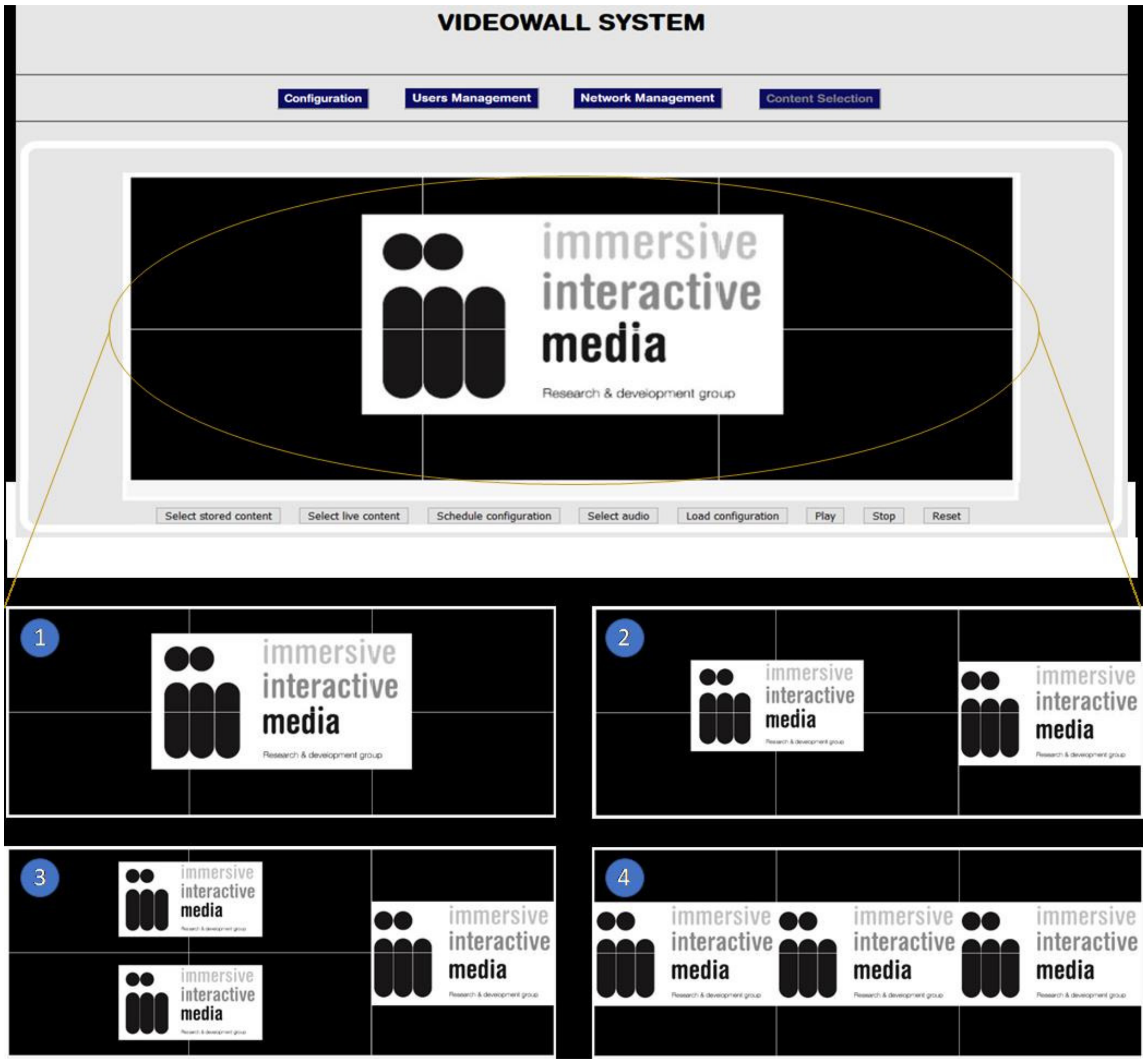


Figure 4. *Videowall configuration and content selection screen (top) and different examples of section configurations for a 2x3 videowall (bottom)*

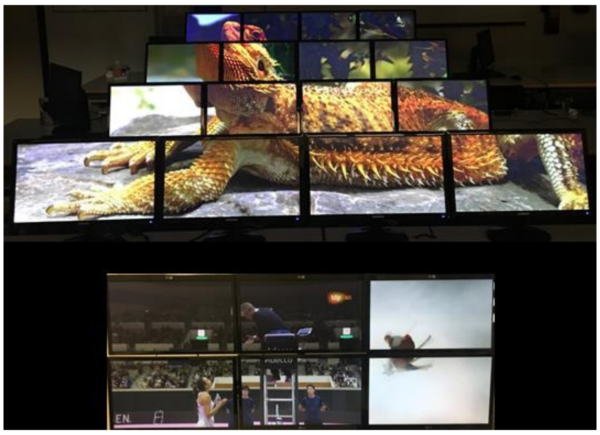


Figure 5. *4x4 (top) and 2x3 (bottom) implemented videowall systems for the objective and subjective evaluation, respectively*

the video player, as RPi devices were selected as *render devices*, two different options were initially considered: the use of the player and sync modules of the *Gstreamer* framework [33] and the use of the *OMXPlayer* player [34]. First, the *Gstreamer* framework was installed in the *render devices*, as the authors already had some experience with this framework in Linux-based systems and with inter-device/destination media sync mechanisms [35]. However, after installing the latest version of the framework, the playout performance was not as good as expected in the *RPis* and authors could not make the videowall system work properly (as improving Gstreamer functionalities for this specific purpose is out of the scope of this work). Secondly, *OMXPlayer* player [34] was tested, and it provided a good performance. Consequently, it has been selected as the video player for the prototype of the proposed videowall system. This player has been designed specifically for the *RPi* device and optimizes the video player's resources by natively using the hardware decoding for H.264 streams. To achieve sync, a Python library [8] has been used to control the playout processes through a communication bus called Desktop Bus [9] (*D-BUS*). One drawback was found to be that the OMXPlayer does not include the option of playing out content with a rotation angle different from 90, 180 or 270 degrees. So, in order to enable this feature in videowalls with warp distribution, an ffmpeg [10] -based script has been developed. This script creates copies of the available content with the necessary rotation angles, which have been previously indicated when customizing and configuring the distribution of the videowall system. This script is run once a videowall system with warp distribution has been set up and new content to be displayed on it is uploaded.

Regarding the image viewer, a Python-based application has been developed. This way, each *render device* gets the selected image and is capable of cropping it and showing the selected region in full screen mode.

Global Time Server & Client As the Global Time Server, the *ntpd* daemon [11] has been installed in the *VCPU*. This way, it can be queried by *render devices* to get a global and common time reference. Regarding the *render devices*, NTP support has to be enabled by installing the package *ntp* [12].

IP Streaming of Live TV Content The*DVBlast open-source network streamer* [13] software has been installed in the *VCPU* in order to receive, decode and stream the live TV broadcasted content. This way, broadcast TV content can be streamed (via multicast RTP/UDP/IP) through the private LAN to the group of *render devices* of the corresponding section. Therefore, in this case, those *render devices* are only required to decode RTP content, which is a feature of the chosen *OMXPlayer*.

Web Server In order to host the VCSM, an Apache [14] web server has been set in the *VCPU*U.

FTP Server The *vsftpd* daemon [15] has been installed in the *VCPU*. After its installation and configuration, authorized users can upload any media content, to specific folders, depending on their privileges or permissions.

Content The selected live content (Spanish DVB-T 1080p TV channels) has a framerate of 25 fps, which implies one frame every 40ms. Additionally, regarding the available stored content, RPi devices support up to 1920x1080 pixels resolution content each, so higher resolution content can be played out in the videowall section if it is previously prepared and cropped in portions with that maximum resolution. Additionally, stored content with different aspect ratio have been selected (e.g., 16:9 for 4x4, 3x3, 2x2 sections, 16:18 for 4x2, 2x1 sections, 24:9 for 2x3 sections or

32:9 for 2x4, 1x2 sections), in order to test that the proposed system automatically fits the media content in the whole displays of each section as long as the aspect ratio is maintained; or places the presented content in the corresponding part of the displays (i.e., left, right, top or bottom side) when it does not fit in all the displays of the whole section.

Data Server In order to store the registered users and the custom configurations of each user, *sqlite3* [16] has been used as the SQL-based server in the VCPU.

Communication Modules To establish the required communication channel, the WebSocket technology [36] has been adopted (as an IETF standard, any modern browser should support this technology [17]).

5.1.3 Implemented Sync mechanism

In the prototypes, an M/S-based sync control scheme and both S&P and AMP playout adjustment techniques have been chosen and implemented just for validation purpose. As mentioned before, the reason for choosing it is that it provides the best performance in terms of scalability, traffic overhead and interactivity (low delays) [18]. Moreover, it does not require any additional module or entity to calculate the achieved sync level, thus allowing to lower the overall cost of the implemented system. As mentioned before, the comparison of different solutions for playout synchronization is out of the scope of the paper. In each video section there will be one *render device* holding the master role and the other *render devices* in that section will hold the slave role.

The implemented sync mechanism in the prototype involves two steps, an initial playout sync and a smooth sync during the playout. The initial sync enables all the *render devices* to start the media playout processes at the same instant. The smooth sync during the playout allows monitoring the asynchronies between the playout processes of the *render devices* and correcting them when needed. These two tasks are performed by the sync manager in each *render device* (Figure 6).

Initial Playout Sync When the involved *render devices* receive a message from the videowall control and management service, indicating the start of video or images playout, this order includes the instant when the playout must start (according to the global time clock reference). Therefore, if all the involved *RPi* have their internal clocks synchronized with the global time clock server, all of them will start its playout at the same time (i.e., synchronously).

Smooth Sync during the Playout As the proposed system can handle both live and (locally or remotely) stored media content, two different sync mechanisms have been designed.

Stored content smooth playout sync

In the previous prototype [8], in order to synchronize the playout processes of each *RPi* and the involved instances of the *OMXPlayer*, a Python-based sync module, called *pyOmxSync* [18], based on the use of a Python-based sync library from a third-party solution, was adopted. However, in order to enhance the stability and accuracy of the initial system, another sync mechanism has been designed in order to be compatible with both stored and live content, as the synchronization achieved by using that Python-based sync module was a bit unstable and was not intended to synchronize playout processes of live content. This new mechanism for the accurate sync of the playout processes of the involved OMXPlayer instances (when connected through IP networks), is carried out by following a M/S sync control scheme.

In each videowall section, there is a *RPi* device acting as the Master (hereafter called *M-RPi*). *M-RPi*periodically sends messages with its current playout timing information (video playout instant and global time). These sync messages are sent with a configurable frequency (e.g., every 1s), by using the multicast IP address and port selected for each section of the videowall (see VIDEO message format in Section II). As soon as the rest of slave RPi devices (in the same videowall section and playing out the same video content) receive the required information from the *M-RPi*, they compare it with their own playout timing information. If the playout asynchrony exceeds a configured threshold (a.k.a. *lower asynchrony threshold*), the slave RPi should correct its playout process with an adjustment. By following the M/S sync control scheme, the number of tiles minimally affect the asynchrony mean values and their occurrence.

In the designed sync mechanism, both S&P and AMP techniques are used. Skips are carried out by slave *render devices* when the calculated asynchrony value is significantly higher than the configured asynchrony threshold. A *higher asynchrony threshold*. A higher asynchrony threshold has been defined (with a value of, e.g., four times the value of the lower threshold), and when exceeded, a skip in the playout process will be carried out in order to get in-sync quickly. This way, noticeable and annoying too long adjustments will be avoided. However, when detected asynchrony levels are between both thresholds, only AMP techniques are adopted to get a smooth sync without skips and pauses, that are more noticeable and annoying.

Just after every adjustment, and once the playout processes have been successfully synchronized, each *RPi* waits a (configurable) *guard period* to sync again, in order to achieve a stable playout process before making another consecutive adjustment. During this guard period, it will ignore any received sync message from the *M-RPi*. Figure 6 presents the flow diagram of the described synchronization process.

Live content smooth playout sync

In order to adjust the different playout processes from each *RPi* and the involved instances of the OMX-Player when the selected content is live, the OMX-Player's source code has been modified to manage playout timing information. Similarly to the stored content scenario, the *M-RPi* sends its playout timing information to the slave *render devices* in the same section via multicast. In this case, in order to achieve sync, only AMP techniques have been adopted. The use of S&P techniques can cause buffer underflow and overflow problems. As the content is live and all the involved *render devices* receive the streaming from the same local source in the LAN (IP streamer in the *VCPU*), async values are not extremely high during the playout, although they are annoying if they are not corrected. They are not high enough to require aggressive adjustment techniques such as S&P as for stored content.

Similar to when playing stored content, after every adjustment, each *render device* waits a (configurable) *guard period* to sync again, discarding any received message from the *M-RPi* during that period.

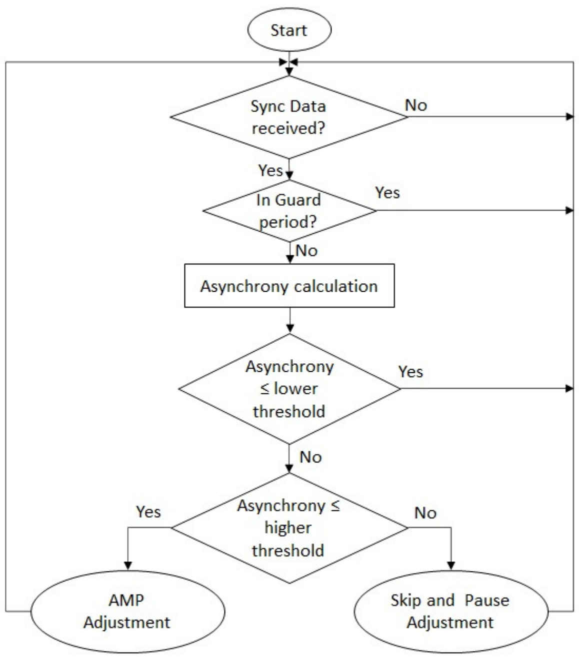


Figure 6. *Flow diagram of the proposed sync mechanism*

Configuration in the implemented prototypes The *lower asynchrony threshold* has been configured to ±60ms to avoid annoying out-of-sync situations between close displays. Several values have been tested and this is the lowest one that has been proven to provide successful accurate sync and stable playout performance when using RPi devices. In a similar way, the playout timing transmission period has been set to 240ms and the guard period after a playout adjustment has been set to 1s. After some tests with different values, both variables have been configured with those values to send enough updated information to the slave *render devices* (transmission period) and to have time enough to stabilize the playout process after an adjustment (guard period). Moreover, the AMP adjustment technique is performed by increasing/decreasing 10% of the playout nominal rate. According to [37, 38], playout rate variations of up to 25% are not noticeable by users. The value of 10% has been chosen after testing several values lower than 25% in the lab. By using that value, playout buffer overflows and overflows are avoided for live content and it has been proven to be fast and accurate enough for both types of contents. Additionally, regarding stored content playout, the S&P adjustment technique is performed only if the asynchrony value exceeds ±180ms (*higher asynchrony threshold*).

5.2 Objective Evaluation

Two different cases have been considered to perform a more rigorous objective evaluation: one involving live content and another one involving stored content. Additionally, in the stored content case, the aforementioned sync approach based on the use of a publicly available sync module, *pyOmxSync*, has also been used, in order to evaluate the performance improvement in the new version of the prototype regarding the one in [8] [19], which did not work for live content. Before measuring the achieved sync level, all the different supported configurations in both 2x3 and 4x4 versions of the videowall (see Figure 1) were visually tested with both stored and live content in all the possible combinations of different sections. Figure 9 shows real photos of some of those tested configurations for the 2x3 version [20], concretely the ones following some of the videowall configurations presented in Figure 4. Particularly in that figure, in 1a) and 1b) a 2x3 section with stored and live (Spanish *Teledeporte* TV channel) content is presented, respectively; in 2) two sections are defined: a 2x2 section with live content (same TV channel) and a 2x1 section with stored content are presented; in 3) two 1x2 sections with stored content and a 2x1 section with live content (same TV channel) are presented; and in 4) three 2x1 sections, one with live content (same TV channel) and the other two sections with stored content, are presented. Additionally, in 5), a videowall with warp distribution is shown with a 2x2 and a 2x1 sections, both with stored content. In these snapshots, it can be seen that in the videowall system the displays (tiles) involved in each video section are in-sync, obtaining a seamless larger display for each section.

Regarding live content, the maximum available quality from the DVB-T (broadcast) provided channels in Spain is 1080p and 25fps. However, regarding the stored content case, an 8K 60fps content has been used in order to test the highest possible quality that the 4x4 videowall can support, as using 4x4 displays implies that each *render device* playouts up to 1080p portions of the content. For such purpose, the 8K content has previously been cropped in 16 portions of exactly 1920x1080 pixels at 60fps each. In particular, that content has been obtained from YouTube [21]. The recorded video available in the link provided in footnote 19 shows the achieved accurate sync in this videowall configuration.

To evaluate the achieved sync level with each sync mechanism, ten 5-minute sessions have been carried out and asynchrony values have been registered during them in all the involved *render devices*.

On the one hand, regarding stored content, when the proposed sync mechanism has been used, the mean asynchrony value has been 22ms, with a 95% confidence interval (C.I.) of ± 2.89ms (i.e., 95% of the registered asynchronies are in between 19ms and 25ms). When using the Python-based sync library from a third-party solution, the mean asynchrony value has been 28ms, with a 95% C.I. of ± 7,5ms. On the other hand, regarding live broadcast content, the achieved average asynchrony has been a bit higher, 41ms, with a 95% C.I. of ± 1.81ms (i.e., 95% of the registered asynchronies are in between 39ms and 43ms). Table II summarizes the obtained results. It can be observed that the proposed system achieves accurate sync for both stored and live content. For stored content the implemented sync solution reduces the mean value of the synchrony by 20% regarding the previously used solution based on the Python-based sync library, and is also more stable (as shown later). These values of asynchrony are practically imperceptible to the human eye, as it can be concluded from the results of the subjective evaluation, which are presented in the next Subsection.

During the sessions, unexpected high asynchrony values have been registered (especially if live content is selected) in very few occasions, mainly due to the playout process fluctuations and to their initial playout start. *OMXPlayer* buffers the received (live) data until there is enough content to start the rendering. It can happen that a *a render device* starts its playout process and right away receives an I-frame. However, another *render device* can start its playout process a bit later, missing that I-frame, and must wait to the next one, which results in a noticeable asynchrony (depending on the Group of Pictures -GOP- length). Therefore, *render devices* may not need to buffer the same amount of content during all the time and this may lead to playout asynchronies. These fluctuations are unexpected and unavoidable, but they are corrected by the proposed sync mechanism as soon as they are detected.

Additionally, measurements for longer sessions have been registered, to test the overall behavior of the

Table 2. *Obtained mean asynchrony values and 95% C.I.*

Type of content	Own sync solution	Third-party solution
Stored	22 (±2.89) ms	28 (±7.5) ms
Live (broadcast DVB-T)	41 (±1.81) ms	Not suitable

proposed system and the evolution of the sync level. In order to show the stability of the playout processes when the sync mechanism proposed in this paper or the 3rd party sync solution are implemented in the videowall, Figure 8a is provided. It presents the evolution of the measured asynchrony values by a *render device* in a 10-minute session, playing out stored content and without playout adjustments. It can be observed how, even during stable states of the playout processes (i.e., when no adjustments are required because the lower asynchrony threshold is not exceeded), the registered asynchrony values fluctuate less when the included sync mechanism is the one proposed in this paper. More (and wider) fluctuations can be annoying and even can lead to false asynchrony detections and thus, forcing a playout adjustment when it should not be needed. Moving average graphs of 10 samples have been added to better observe the evolution in time of the asynchrony when both mechanisms are implemented.

Additionally, Figure 8b shows the evolution of the measured asynchrony in one of the 16 *render devices* during a 20-minute session playing out live content and with playout adjustments due to the sync mechanism proposed in the paper. In order to not enlarge the paper with many similar graphs, only this case has been selected because it shows the worst sync performance. It can be seen how some adjustments are required (red crosses) to keep the asynchrony values lower than the *lower asynchrony threshold* (±60ms). Note that adjustments in that figure coincide with the detected (and corrected) asynchrony values exceeding that threshold. Actually, the detection of those values of asynchrony exceeding the threshold triggered the corresponding adjustment processes. In any case, the asynchrony evolution confirms that the designed sync mechanism behaves correctly. In this case, also a moving average graph of 10 samples has been added (black continuous line) to better observe the evolution in time of the measured asynchrony.

Figure 8c presents the obtained cumulative frequency distribution (CFD) graph for the three cases: live content with the proposed sync mechanism, stored content with the proposed sync mechanism and stored content with the 3rd party sync solution. It can be observed how asynchrony values are distributed in terms of probability of occurrence. Asynchrony values are lower in the stored content case (even lower with the use of the own sync mechanism) compared to the live content case, which is logical and expected. For instance, on the one hand, when the stored content is selected with the proposed sync mechanism, there is a 90% of probabilities of having asynchronies of, as maximum, 35ms. Additionally, if the 3rd party sync solution is adopted, there is a 90% of probabilities of registering a maximum asynchrony of 40ms. On the other hand, when live content is selected, the probability of having, as maximum 40ms, decreases to 70%, although this is still a satisfactory result.

In conclusion, as seen in this Section, the performance of the proposed sync mechanisms in the proposed videowall system can be considered as accurate enough. The configured *lower asynchrony threshold* value has been proven to be low enough to keep playout processes within acceptable asynchrony levels and without negatively affecting the playout processes of *render devices*. Fluctuations (or unexpected issues) have been corrected as soon as they have been detected.

5.3 Subjective Evaluation

In order to subjectively assess the proposed system, a 2x3 videowall has been implemented and temporarily located in four different places at the authors' University Campus (one week in each): the information point, the sports office, the management office and the library hall. In each place, it has been used by the workers in those places as an additional tool to provide information to the students in the Campus. Figure **??** shows images of the videowall system placed in those four locations.

Specifically, up to 15 participants (composed of workers from the University Campus, with no rela-

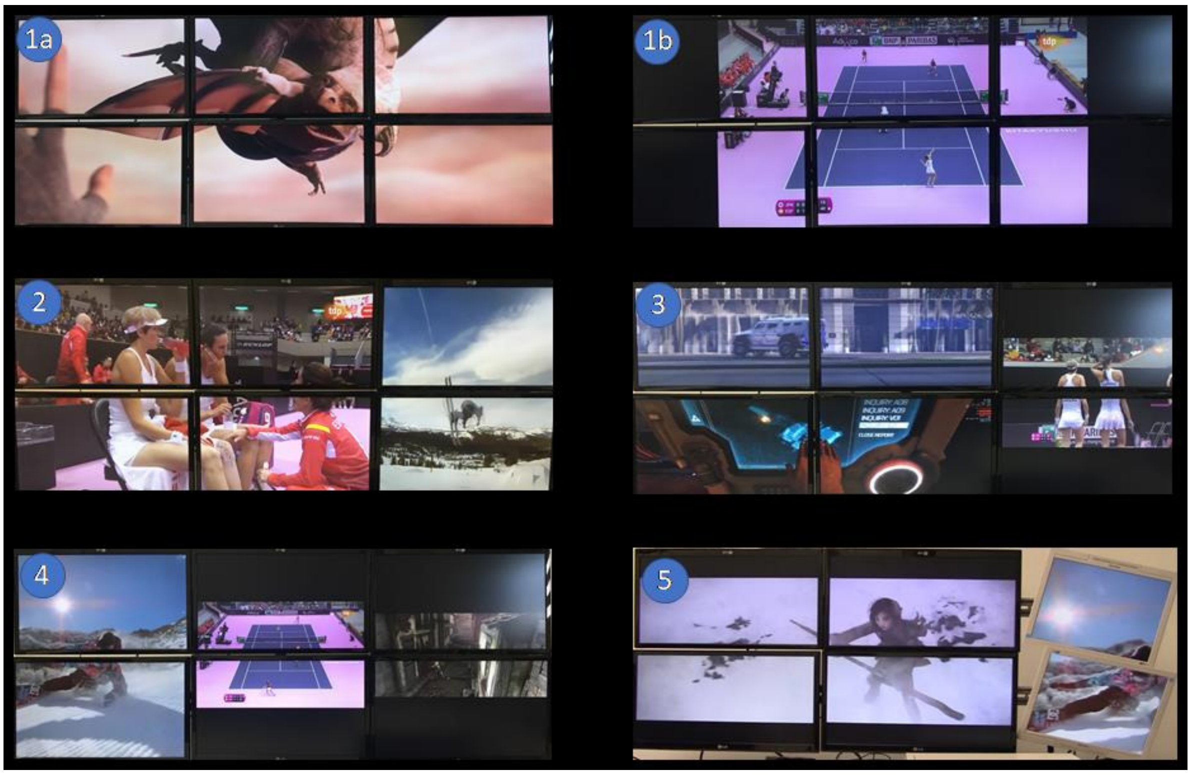


Figure 7. *Images (1-4) of the 2x3 videowall in mosaic distribution following the four section configurations shown in Figure 4 and an image (5) of the videowall in warp distribution as in the bottom part of Figure 3*

tion with the work presented in this paper) have used the videowall system for one week. 62% of them were women. Regarding their age, 23% of the participants are between 30-39 years old, 54% between 40-49 years old, and the remaining 23% between 50-59 years old. Only a 30% of the participants have a technical profile.

Regarding the use of the system, after a 30-minute training period, they have been able, for one week, to modify the video sections, the playout processes, the content selection, to define playlists, etc., as ordinary users. After each week, the participants in each place, have filled a usability questionnaire and have answered some questions about the perceived level of asynchrony.

In order to validate the usability of the web application, the System Usability Scale (SUS) test, described in [39], has been used. It provides a score from 0-100 points, being 0 points the worst imaginable usability score and 100 points the best imaginable usability score, as defined in [40]. The following list shows the 10 questions of the SUS questionnaire.

- Q1. I think that I would like to use this system frequently.
- Q2. I found the system unnecessarily complex.
- Q3. I thought the system was easy to use.
- Q4. I think that I would need the support of a technical person to be able to use this system.
- Q5. I found the various functions in this system were well integrated.
- Q6. I thought there was too much inconsistency in this system.
- Q7. I would imagine that most people would learn to use this system very quickly.
- Q8. I found the system very cumbersome to use.
- Q9. I felt very confident using the system things before I could get going.
- Q10. I needed to learn a lot of with this system.

Figure 8d summarizes the obtained results. After analyzing the responses, the obtained usability score has been of 88.65 with a standard deviation value of 7.68, which can be labelled as excellent according to [40]. Thus, it can be stated that the web application has covered the requirements to be easy to use and intuitive.

Moreover, participants were also asked to score the perceived level of sync by using the Mean Opinion Score (MOS) [41], and a 5-level Likert-type scale, where rating 1 means that the system is not synchronized at all and 5 means that the system is perfectly synchronized. The obtained MOS regarding the perceived sync has been of 4.3 ± a 95% confidence interval of 0.261, which is a very satisfactory value. Notice that every time the playout in each section of the videowall is started there is a short initial transitional period in order to achieve the accurate synchronization state between displays. During that period there can exist an asynchrony that can be noticeable for participants and might have affected their overall sync perception.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a scalable hardware and software architecture for tiled display systems, which can be implemented by using low-cost devices (i.e., affordable systems), together with a dynamic web-based management and configuration service have been proposed, implemented (including an own sync mechanism) and tested both objectively and subjectively. Mosaic or warp distributions can be remotely and dynamically configured through a user-friendly web application, and for both stored and live content. The embedded web-based configuration and management service allows a multi-platform/device access to the configuration of the complete system.

Appropriate cheap devices and modules can be selected to get a low-cost system. In order to test the proposed system, two RPi-based videowall prototypes have been implemented and both objective and subjective evaluations have been carried out, obtaining very satisfactory results. On the one hand, the objective evaluation has been conducted over a 4x4 mosaic videowall with 8K stored content and DVB-T live content. The obtained results confirmed the good performance of the implemented own sync mechanism (outperforming another 3rd party solution) and of the overall system. On the other hand, the subjective evaluation has been conducted by using a portable 2x3 videowall located in four different places at the authors' University Campus. The usability of the configuration service has been rated as excellent and the perceived asynchrony has been scored as very satisfactory by the participants in the subjective assessment.

As future work, this videowall system is going to be continuously revised and updated. Regarding the possibility of playing remote broadband video content, authors expect to add the support for HTTP-based adaptive streaming technologies (such as DASH [42] or HLS [43]) in future versions. Regarding audio contents, as mentioned, the inclusion of multiplexers will be studied in order to allow that the different audio streams (e.g., the associated audio content of each section of the videowall) can be heard through different output devices (e.g., speakers, wired or wireless headphones, etc.), independently. Moreover, currently, model 4 of the RPi device is already commercially available, with better features and video outputs being able to render up to 4K video content. They will be used in future prototypes in order to render higher resolution videos (e.g., 16K) with less displays and better performance.

NOTES

[1]https://www.userful.com/ (last access: July 2022)
[2]http://www.samsung.com/es/business/solutions-services/smart-signage-solutions/smart-signage-solutions/magicinfo-videowall (last access: July 2022)
[3]http://www.lg.com/us/business/commercial-display/displays-tvs/video-walls (last access: July 2022)
[4]In this paper, implemented videowalls using one or few computers, each with several embedded graphic cards (with several outputs connected to different displays), are not considered.
[5]In this paper, implemented videowalls using one or few computers, each with several embedded graphic cards (with several outputs connected to different displays), are not considered.
[6]Raspbian is a simplified version of Debian, designed to run under ARM processors. https://www.raspberrypi.org/downloads/raspbian/ (last access: July 2022
[7]Raspberry Pi 3 is a low-cost microcomputer that offers, among other features: a Fast Ethernet interface, 802.11n WiFi, open-source software and hardware decoding of H264 content with 1080p

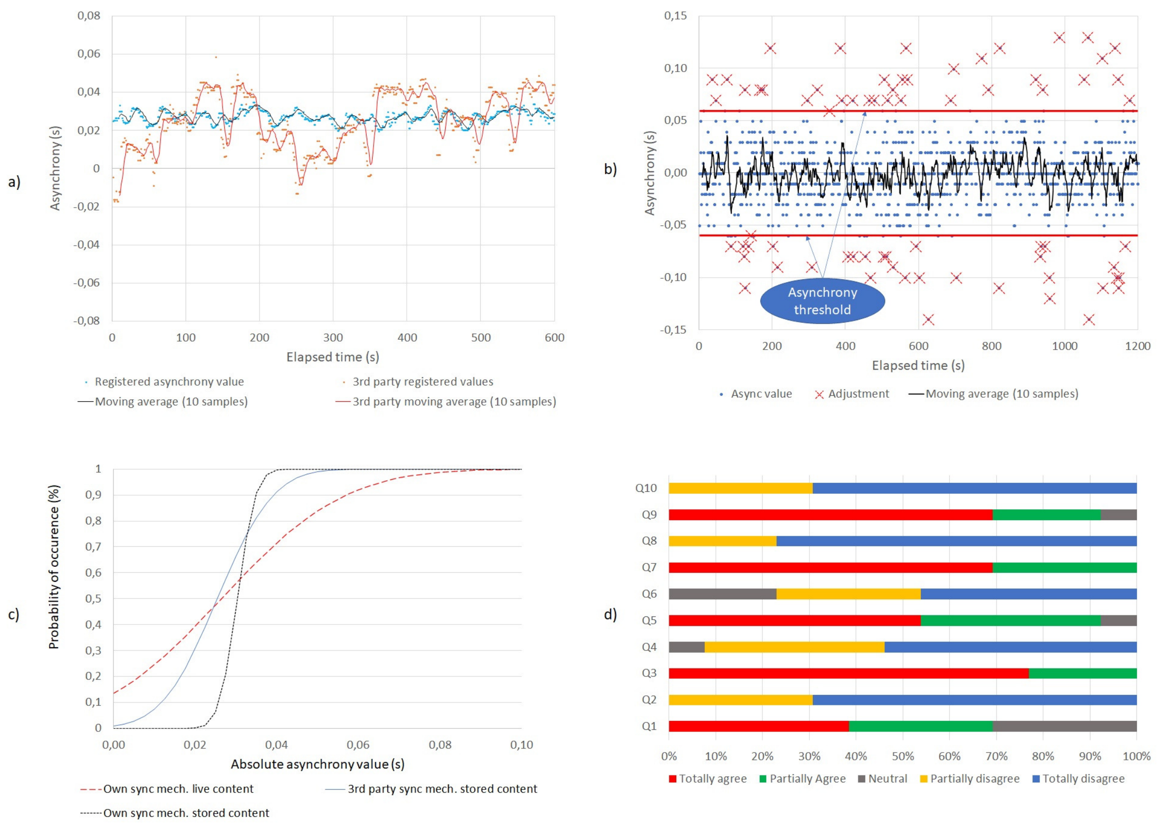


Figure 8. *Evaluation results*

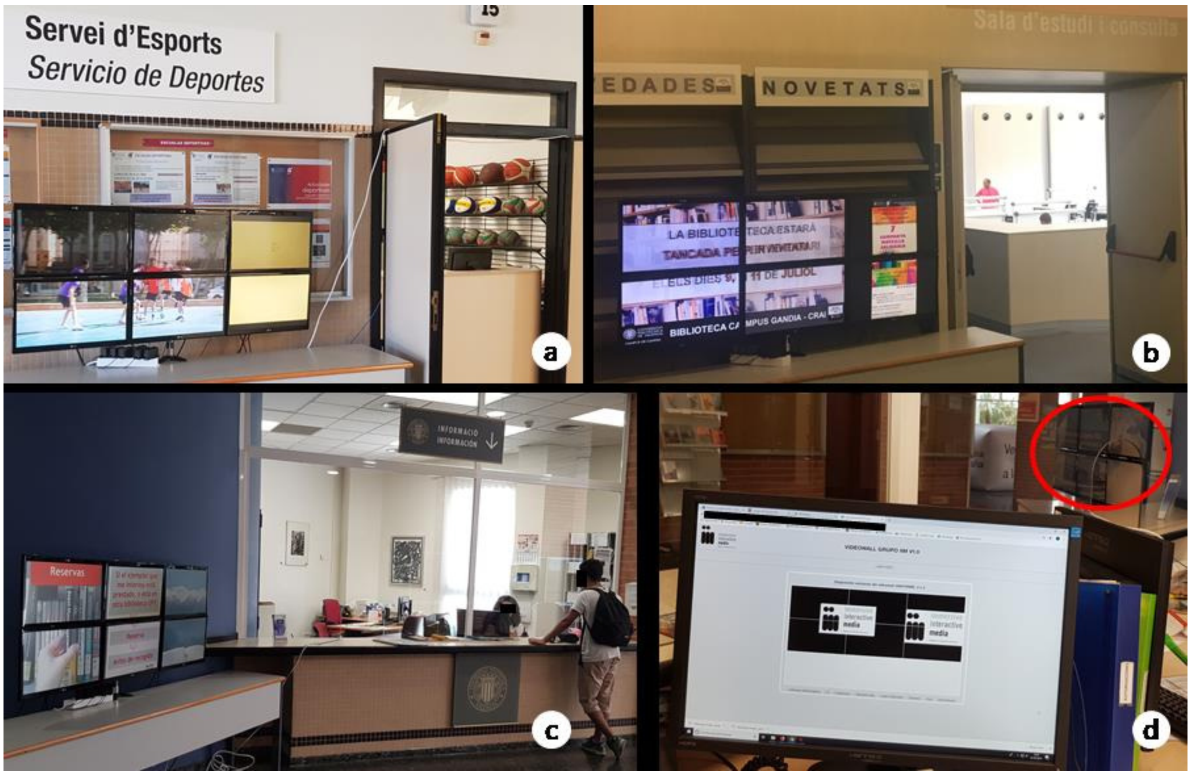

Servei d'Esports
Servicio de Deportes
a
b
c
d
NOVETATS
LA BIBLIOTECA ESTARÀ TANCADA PER INVENTARI
ELS DIES 9 I 11 DE JULIOL
BIBLIOTECA CAMPUS GANDIA - CRAI
Reservas
VIDEOWALL GRUPO IM V1.0
interactive media

Figure 9. *The 2x3 videowall placed in the sports office (a), the library hall (b), the information point (c) and the management office (d) from the participants point of view*

resolution content.

[8]OMXPlayer for Pyhton. Official Repository: https://github.com/willprice/python-omxplayer-wrapper (last access: July 2022)

[9]https://www.freedesktop.org/wiki/Software/dbus/ (last access: July 2022)

[10]FFmpeg. Official site: https://ffmpeg.org/ (last access: July 2022)

[11]http://manpages.ubuntu.com/manpages/trusty/man8/ntpd.8.html (last access: July 2022)

[12]https://packages.debian.org/stretch/ntp (last access: July 2022)

[13]https://www.videolan.org/projects/dvblast.html (last access: July 2022)

[14]https://httpd.apache.org (last access: July 2022)

[15]http://manpages.ubuntu.com/manpages/bionic/man8/vsftpd.8.html (last access: July 2022)

[16]http://manpages.ubuntu.com/manpages/bionic/man1/sqlite3.1.html (last access: July 2022)

[17]https://caniuse.com/#feat=websockets (last access: July 2022)

[18]https://github.com/markkorput/pyOmxSync (last access: July 2022)

[19]In this paper, regarding sync, the author's purpose is only to show that the newly implemented own sync solution performs correctly and in a stable way, and that it is better than the one included in the previous prototype (based on the publicly available sync module, pyOmxSync). The comparison with the accuracy or stability of the sync solutions included in other more professional or commercial products is out of the scope of this paper.

[20]Photos of the 4x4 videowall have not been provided in that figure because for that distribution the displays were not fixed to any structure. Each row of 4 screens were put on a table with different height (lower/higher height for the first/fourth row) in a lab and watched and recorded the view from a specific position to see all the screens as similar as possible to a videowall (see the top image in Figure 5 and the demo video in https://bit.ly/2JKzh8T, last access: July 2022)

[21]https://youtu.be/ku3wCvIUdHQ, last access: July 2022)

References

[1] RIVERA L P, VERGARA S V C, VIVEROS A M. Visual data mining over a video wall[C]// CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers. IEEE, 2012: 239-244.

[2] VAN DER WERF J M E, DE FEIJTER R, BEX F, et al. Facilitating collaborative decision making with the software architecture video wall[C]// 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017: 137-140.

[3] YIN P, JIANG X, SHI J, et al. Multi-screen tiled displayed, parallel rendering system for a large terrain dataset.[J]. Int. J. Virtual Real., 2006, 5 (4): 47-54.

[4] NODA S, EBARA Y, ISHIDA T, et al. Implementation of high presence video communication system for multiple users using tiled display environment[C]//2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops. IEEE, 2015: 494-499.

[5] NI T, SCHMIDT G S, STAADT O G, et al. A survey of large high-resolution display technologies, techniques, and applications[C]//IEEE Virtual Reality Conference (VR 2006). IEEE, 2006: 223-236.

[6] BUNDULIS R, ARNICANS G, ALBERTAS C. Architectural and technological issues in the field of multiple monitor display technologies [C]//Frontiers in Artificial Intelligence and Applications. Databases and Information Systems VII: Selected Papers from the Tenth International Baltic Conference, DB&IS 2012: volume 249. 2013: 317-329.

[7] RIDOUX J, VEITCH D. Principles of robust timing over the internet[J]. Communications of the ACM, 2010, 53(5): 54-61.

[8] SALVADOR P, BORONAT F, MONTAGUT CLIMENT M A, et al. Sistema videowall de bajo coste basado en raspberry pi, personalizable y configurable dinámica y remotamente vía web[J]. XIII Jornadas de Ingeniería telemática (JITEL 2017). Libro de actas, 2018: 318-325.

[9] MONTAGUD M, BORONAT F, ROIG B, et al. How to perform amp? cubic adjustments for improving the qoe[J]. Computer Communications, 2017, 103: 61-73.

[10] NAVEEN K, VENKATRAM V, VAIDYA C, et al. Sage: the scalable adaptive graphics environment[J]. 2004.

[11] MARRINAN T, AURISANO J, NISHIMOTO A, et al. Sage2: A new approach for data intensive collaboration using scalable resolution shared displays[C]//10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing. IEEE, 2014: 177-186.

[12] DOERR K U, KUESTER F. Cglx: a scalable, high-performance visualization framework for networked display environments[J]. IEEE transactions on visualization and computer graphics, 2010, 17(3): 320-332.

[13] HUMPHREYS G, HOUSTON M, NG R, et al. J, t. klosowski," chromium: a stream-processing framework for interactive rendering on clusters [C]//ACM Conference on Computer graphics and interactive techniques. 2002: 693-702.

[14] JOHNSON G P, ABRAM G D, WESTING B, et al. Displaycluster: An interactive visualization environment for tiled displays[C]//2012 IEEE International Conference on Cluster Computing. IEEE, 2012: 239-247.

[15] BUNDULIS R, ARNICANS G. Infiniviz: Taking quake 3 arena on a large-scale display system to the next level[C]//2018 23rd Conference of Open Innovations Association (FRUCT). IEEE, 2018: 91-98.

[16] JIMENEZ HERRERO A. Videowall disforme sobre redes ip[D]. Universitat Politècnica de Catalunya, 2016.

[17] BORONAT F, LLORET J, GARCIA M. Multimedia group and inter-stream synchronization techniques: A comparative study[J]. Information Systems, 2009, 34(1): 108-131.

[18] MONTAGUD M, BORONAT F, STOKKING H, et al. Inter-destination multimedia synchronization: schemes, use cases and standardization[J]. Multimedia systems, 2012, 18(6): 459-482.

[19] KWANG-YONG K, CHANG-WOO Y, WON L. The design of server virtualization based video-wall control system[C]//2012 7th International Conference on Computing and Conver-

gence Technology (ICCCT). IEEE, 2012: 401-404.

[20] W3C. Synchronized Multimedia Integration Language SMIL[Z].

[21] BUNDULIS R, ARNICANS G. Concept of virtual machine based high resolution display wall [C]//2014 IEEE 2nd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE). IEEE, 2014: 1-6.

[22] GOODYEAR A, HOGBEN C, STEPHEN A. Pi-Wall[Z].

[23] ALABDULSALAM B A, ALSALMAN F A, ALSHAKHS Z H, et al. Dynamic video wall tile creation using raspberry pi3[C]//2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS). IEEE, 2017: 268-271.

[24] MIHALOPOULOS V, SVERONIS K. Creating a videowall with yodeck[Z].

[25] SHIN I, LEE S, LEE E, et al. S/w based frame-level synchronization for irregular screen processing system[J]. ETRI Journal, 2016, 38(5): 868-878.

[26] KANDA J, TSUBAKI Y, YOSHIDA H, et al. A multiple terminal synchronous display method for ultra-high resolution display systems[C]// 2014 IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2014: 510-511.

[27] VAN BRANDENBURG R, VEENHUIZEN A. Immersive second-screen experiences using hybrid media synchronization[C]//MediaSync Workshop. 2013: 1-7.

[28] HOWSON C, GAUTIER E, GILBERTON P, et al. Second screen tv synchronization[C]// 2011 IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin). IEEE, 2011: 361-365.

[29] BROWN A, EVANS M, JAY C, et al. Hci over multiple screens[M]//CHI'14 Extended abstracts on human factors in computing systems. 2014: 665-674.

[30] D. MILLS J B, J. Martin, KASCH W. Network Time Protocol Version 4: Protocol and Algorithms Specification: 5905[R/OL]. 2010. https://www.rfc-editor.org/rfc/rfc5905.txt.

[31] EIDSON J C, FISCHER M, WHITE J. Ieee-1588™ standard for a precision clock synchronization protocol for networked measurement and control systems[C]//Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting. 2002: 243-254.

[32] SCHULZRINNE H, CASNER S, FREDERICK R, et al. Rtp: A transport protocol for real-time applications[R]. 2003.

[33] GStreamer: open source multimedia framework [Z].

[34] OMXPlayer Project[Z].

[35] MARFIL D, BORONAT F, SAPENA A, et al. Synchronization mechanisms for multi-user and multi-device hybrid broadcast and broadband distributed scenarios[J]. IEEE Access, 2018, 7: 605-624.

[36] FETTE I, MELNIKOV A. The websocket protocol: 6455[R/OL]. 2011. https://www.rfc-editor.org/rfc/rfc5905.txt.

[37] CHUANG H C, HUANG C, CHIANG T. Content-aware adaptive media playout controls for wireless video streaming[J]. IEEE Transactions on Multimedia, 2007, 9(6): 1273-1283.

[38] SU Y F, YANG Y H, LU M T, et al. Smooth control of adaptive media playout for video streaming[J]. IEEE transactions on multimedia, 2009, 11(7): 1331-1339.

[39] BROOKE J, et al. Sus-a quick and dirty usability scale[J]. Usability evaluation in industry, 1996, 189(194): 4-7.

[40] BANGOR A, KORTUM P, MILLER J. Determining what individual sus scores mean: Adding an adjective rating scale[J]. Journal of usability studies, 2009, 4(3): 114-123.

[41] Methods for subjective determination of transmission quality[Z]. 1996: 800.

[42] ISO/IEC 23009-1:2014 - Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats.[Z].

[43] PANTOS R, MAY W. HTTP Live Streaming, IETF draft[Z]. 2017.

Biographies

Daniel Marfil was born in Gandia (Spain) and studied Informatics Technical Engineering BSc degree (2011), Telecommunications BSc degree (2015) and Telecommunication Technologies, Systems and Networks MSc (2016) in Universitat Politecnica de Valencia (UPV, Spain). He is a PhD student and an assistant researcher and developer in the Immersive Interactive Media R&D Group. His main topics of interest are communication networks, code developing and media synchronization. He is the author of one book chapter and several research and conference papers.

Fernando Boronat was born in Gandia (Spain), and went to the UPV in Spain, where he studied Telecommunications Engineering. He received the M.E. and Ph.D. degrees in telecommunication engineering from the UPV in 1994 and 2004, respectively. After working for several Spanish telecommunication companies, he moved back to the UPV in 1996. Currently he is an Assistant Professor in the Communications Department. Dr. Boronat is the Head of the Immersive Interactive Media R&D Group (http://iim.webs.upv.es) at the Gandia Campus of UPV. His main topics of interest are communication networks, multimedia systems, multimedia protocols, and media synchronization. He is the author of two books, several book chapters, an IETF RFC and more than 100 research papers. He is Editor of "MediaSync: Handbook on Multimedia Synchronization" (Springer, 2018). He is involved in several IPCs of national and international journals and conferences. He is member of IEEE and ACM.

Fco. Javier Pastor received the Bachelor of Fine Arts degree from the UPV in 1995, the master's degree in industrial design, specializing in design management from Domus Academy, Milan, Italy, and the Doctorate in Fine Arts degree, specializing in design in 2004. From 1997 to 2000, he taught with the School of Design, Cardenal Herrera Oria University CEU San Pablo de Valencia. He was an Industrial Designer, projecting for Olivaterra, Ona Iluminación, NK Electrónica. As a Graphic Designer, he collaborates with companies and public entities generating corporate image manuals and communication strategies for: Sedarías Camilo Miralles, NK Electrónica. From 1998 to 1999, he was an Industrial Designer and a Researcher with Fujitsu Industrial Design Laboratory, Kawasaki, Japan, conceptualizing and designing consumer electronics and mobile telephony products. During this period, he presented the Fujiutsu domestic PC "at ease." In 2000, he joined the Polytechnic University of Valencia, as a Professor with the Drawing Department, where he has been a part of the research group IGD (Research and Design Management) since 1995. From this platform, he works directly with companies in the habitat design sector (Gamamobel, Mobenia, Expormim, and Poalgi), managing product portfolios and establishing communication strategies. He has participated in numerous research projects on design trends, from which publications and articles have been derived. Since 2015, he has also been a part of the Immersive Interactive Media Research and Development Group.

Anna Vidal holds a phD in Mathematics, focused on the area of fuzzy topology research and its applicability. Currently, she is an Assistant professor at UPV and is part of the IIM R&D research group focusing her research on the design, modeling and analysis of multimedia synchronization techniques.