Improving response time interval in networked event-based mulsemedia systems

Estêvão Bissoli Saleme
Federal University of Espírito Santo
Vitória, ES, Brazil
estevaobissoli@gmail.com

Celso A. S. Santos
Federal University of Espírito Santo
Vitória, ES, Brazil
saibel@inf.ufes.br

Gheorghita Ghinea
Brunel University London
London, UK
george.ghinea@brunel.ac.uk

ABSTRACT

Human perception is inherently multisensory involving sight, hearing, smell, touch, and taste. Mulsemedia systems include the combination of traditional media (text, image, video, and audio) with non-traditional ones that stimulate other senses beyond sight and hearing. Whilst work has been done on some user-centred aspects that the distribution of mulsemedia data raises, such as synchronisation, and jitter, this paper tackles complementary issues that temporality constraints pose on the distribution of mulsemedia effects. It aims at improving response time interval in networked event-based mulsemedia systems based upon prior findings in this context. Thus, we reshaped the communication strategy of an open distributed mulsemedia platform called PlaySEM to work more efficiently with other event-based applications, such as games, VR/AR software, and interactive applications, wishing to stimulate other senses to increase the immersion of users. Moreover, we added lightweight communication protocols in its interface to analyse whether they reduce network overhead. To carry out the experiment, we developed mock applications for different protocols to simulate an interactive application working with the PlaySEM, measuring the delay between them. The results showed that by pre-processing sensory effects metadata before real-time communication, and selecting the appropriate protocol, response time interval in networked event-based mulsemedia systems can decrease remarkably.

CCS CONCEPTS

• **General and reference** → **Measurement**; *Experimentation*; • **Computer systems organization** → *Real-time system architecture*; • **Software and its engineering** → *Message passing*; *Publish-subscribe / event-based architectures*; *Software performance*;

KEYWORDS

Mulsemedia systems; Sensory effects; Event-based applications; Multisensory experiences; Evaluation.

ACM Reference Format:
Estêvão Bissoli Saleme, Celso A. S. Santos, and Gheorghita Ghinea. 2018. Improving response time interval in networked event-based mulsemedia systems. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3204949.3204965

1 INTRODUCTION

The advent of novel technologies and innovative devices has allowed the conception of systems in which new ways of interactions are offered to users. Applications which were seen as sheer scientific fiction not so long ago have started becoming viable and being experimented by computer science researchers. Mulsemedia [8, 9] systems fit this evolutionary process by including the combination of traditional media (text, image, video, and audio) with non-traditional ones that stimulate other senses beyond sight and hearing, such as smell, touch, and taste.

A mulsemedia application has different characteristics to traditional multimedia ones. Whilst the latter is dominated by requirements stemming from the continuous, temporal nature of audio and video, mulsemedia applications are ones which integrate these classic media types with non-traditional ones, such as olfactory, haptic (wind) and gustatory. The tendency of these media types to linger (in the atmosphere or on the tongue), the fact that one has little control on the directionality of the sensation, means that issues such as distribution, temporality, production and rendering take on totally new valences in the context of mulsemedia systems. Thus, different multi-sensory experiences require different response time.

Although many studies have found synchronisation threshold times for distinct kinds of sensory effects in different setups such as [1, 10, 14–16, 24, 25, 27], it is still hard to pinpoint precise numbers as straight marks for acceptable delays because it can vary depending on the context of the interaction. For instance, haptic delay perception might even be different in discrete and continuous events in many applications depending on how the operator issues action commands and what information is fed back [19]. Regardless, it is noticeable that delay is more sensitive to some kinds of effects than others and it requires prompt responses. Therefore, identifying temporal thresholds that permeates the mulsemedia chain is valuable from the perspective that it provides developers with guidelines for the design of responsive solutions.

Whilst work has been done on some user-centred aspects that the distribution of mulsemedia data raises, such as synchronisation and jitter, this paper tackles complementary issues that temporality constraints pose on the distribution of mulsemedia effects. Specifically, this paper aims at improving response time interval in networked event-based mulsemedia systems based upon the findings of Saleme et al. [20]. The authors evaluated the integration between a gestural interactive application and a distributed mulsemedia platform called PlaySEM [21] for rendering sensory effects such as wind, lighting, and vibration, specified in MPEG-V, in the users' environment. This platform allows developers to add multi-stimuli in their own multimedia applications by reusing the PlaySEM Sensory Effects Renderer (SER), the central component of the PlaySEM architecture. However, its implementation also raises questions and its authors highlighted four major aspects needed to improve response time interval in networked event-based mulsemedia systems: (i) repetitive actions of sending metadata scripts in real time; (ii) high number of exchanged messages; (iii) use of wireless network; and (iv) use of suitable programming languages for processing sensory effect metadata described in MPEG-V. Concretely, we focus on aspects (i) and (ii) in this paper.

To carry out the experiment, we redesigned the communication strategy of the PlaySEM SER in order to work more efficiently with other event-based applications, such as games, VR/AR software, and interactive applications. Furthermore, we added support to other commonly used lightweight communication protocols recurrently used in CPS (Cyber-Physical Systems) such as CoAP (Constrained Application Protocol) and MQTT (Message Queue Telemetry Transport), and Websocket, apart from UPnP (Universal Plug and Play) which is natively supported by the PlaySEM SER, to analyse whether they reduce network overhead. For evaluating the performance, we developed mock applications for each added protocol to simulate an interactive application working with the PlaySEM SER over a network, measuring the delay between them. The results showed that by pre-processing sensory effects metadata before real-time communication, and selecting the appropriate protocol, response time interval in networked event-based mulsemedia systems can decrease remarkably.

This paper is organised as follows. Section 2 discusses related work. Section 3 depicts the new architecture of the solution, describing its elements, as well as its behaviour in terms of exchanging mulsemedia metadata and events through the network. In Section 4, we describe our experiment design and setup. Section 5 shows the results and raises some discussion contrasting prior findings to this one. Finally, Section 6 presents our final considerations.

2 RELATED WORK

A mulsemedia system can work in two ways: (i) timeline or (ii) event-based. In timeline mode, physical devices (or actuators), which deliver sensory effects to stimulate other senses, are synchronised with a continuous medium in the virtual world, i.e. movies, songs, and so on. In event-based mode, actuators can be activated by events that occur in the virtual world, such as an explosion in a game, or as a response to a stimulus from the real world captured through sensors by the virtual world. Ordinary issues in this kind of system are inextricably linked to the transmission, production and presentation of multiple signals which increase the complexity of developing mulsemedia systems considerably [27].

Instances of timeline mulsemedia systems are SEMP [26], Sensible Media Simulator [11] and Multimedia Multisensorial 4D Platform [3], whereas Sensorama [4] and PlaySEM [21, 22] are instances of both timeline and event-based ones. What all these mulsemedia systems have in common is that they support metadata described in MPEG-V as an input which is processed and then converted into commands for different actuators to render the effects in the users' environment.

Created by Cho [4], Sensorama aims at supporting both timeline and event-based sensory effects when proposing the use of sensory effect metadata combined with a list of events in which can be triggered on demand. Apart from other mulsemedia solutions, it is centred on 4D platforms for movie theatres, and their actuators are part of a CAVE (Cave Automatic Virtual Environment) composed of professional apparatus. Notwithstanding, its pre-defined set of sensory effects created by the author are embedded in the application, which restrains its expansion, not allowing, for instance, the use of other event-based application combined with it.

Saleme and Santos [21] envisaged and implemented a networked scenario where users could take advantage of sensory effects in their own multimedia applications by integrating them to a distributed mulsemedia platform. Following that, Saleme et al. [20] evaluated an integration between an interactive application where users interact by gestures with the PlaySEM platform. In contrast to local applications, networked ones introduce a delay due to the payload's exchange. Though the authors did not carry out user evaluations, they drew attention to crucial technical aspects such as network delay and processing time which could impact the QoE (Quality of Experience) of users. For Eg and Raaen [6], network and computational limitations can indeed interfere with the realisation of users' expectation. Accordingly, in the context of remote delivery of mulsemedia components, issues such as delay, jitter, and synchronisation have been indicated as a research challenge in mulsemedia applications [13].

The findings pointed out in [20] revealed an elapsed time of 27ms to 67ms on average for the whole process of integrating a gestural interactive application to the PlaySEM, covering recognition, packaging, transmission, metadata processing, and execution time (exact point in which the devices are activated). As mentioned in the Introduction (Section 1), the authors highlighted four issues that such applications posed, but also suggested that a CPS approach could be an alternative in this context.

Indeed, it is worth noticing that the arrangement of the environment in [20] had similar characteristics to CPS. According to [17], CPS are "*engineered systems that are built from, and depend upon, the seamless integration of computational and physical components.*" The authors dealt with two systems that integrated multimodal sensing, processing, communication, and controlling, into a physical environment and connected them to the cyber world. Therefore, it is possible to consider that multimedia/mulsemedia and CPS share many common features when physical processes and computations affect each other. It allows us to take the benefits of a mixed approach, taking into account some of the issues that have been discussed in CPS and also mulsemedia systems. In fact, Duchon et al. [5] have approached multimedia systems from this perspective, calling them Cyber-Physical Multimedia Systems (CPMS).

3 ARCHITECTURE AND COMMUNICATION STRATEGY

Once this paper works towards the findings of Saleme et al. [20], we review what the PlaySEM platform and the gestural interactive application are within this Section. Moreover, we describe what changes were made to improve the response time interval. It includes a modification in the PlaySEM's architecture to support other communication protocols, and in its operation mode in order to reduce repetitive tasks and the number of exchanged messages through the network, thus creating a new communication strategy. We refer to that gestural interactive application as GIA to distinguish it from the general term "interactive application" that could be whatever application which accepts any kind of input from humans as it runs.

3.1 PlaySEM and GIA

PlaySEM[1] is an open source platform written in Java and has three main decoupled components: (i) the Sensory Effects (SE) Video Player, (ii) the Sensory Effects Renderer (SER), which processes MPEG-V metadata and prepares commands to control the devices, and (iii) the Microcontroller module, responsible for receiving the aforementioned commands and driving different actuators. The PlaySEM SER is the central subsystem when it comes to dealing with processing sensory effects. It can be used not only by the SE Video Player but by several different applications, from multimedia players to any event-based application such as games, VR/AR software, and interactive applications. To make it possible, the PlaySEM SER provides a UPnP interface through the network to allow other applications to interact with it.

By taking advantage of the PlaySEM SER facilities, Saleme et al. [20] integrated a GIA with the PlaySEM SER. In order to bind the applications and create a new interactive system, they used two out of several services supplied by the PlaySEM SER: *SetSem*, to receive and process MPEG-V metadata scripts, and *SetPlay*, to indicate that a previously processed script can be executed giving rise to sensory effects. Figure 1 depicts the original communication strategy.

The original communication strategy works as follows. After capturing a gesture with the Kinect (RGB-D sensor), the application recognises it (*Recognition task*). Then, the GIA packages the MPEG-V metadata script to be sent to the PlaySEM SER UPnP Service (*Packaging task*). The message *SetSem* containing an MPEG-V script related to the recognised gesture is then transmitted through the network (*Transmission task*). Subsequently, the PlaySEM SER Controller converts the received MPEG-V script into commands for handling the physical devices (*Metadata processing task*). Following that, the PlaySEM SER sends a confirmation message to the GIA to signal that the script was processed and can be executed (*Transmission task*). After receiving it, the GIA transmits the message *SetPlay* to indicate that the execution can start (*Transmission task*). Finally, the script is executed (*Execution task*) and then sensory effects are rendered in the user's environment (*Rendering task*).

3.2 Service layer and communication protocols

Compared to local applications whereby processing is only on one side, networked ones insert a delay due to message exchanging. On the one hand, provided that developers use the PlaySEM SER, they do not have to be concerned about rendering sensory effects. On the other hand, the number of messages and the way they are exchanged over the network can be an issue. Bearing this in mind, we expanded the architecture of the PlaySEM SER to support different ways of communications in terms of: (i) reducing the number

[1]PlaySEM available at: https://github.com/estevaosaleme/PlaySEM_SERenderer

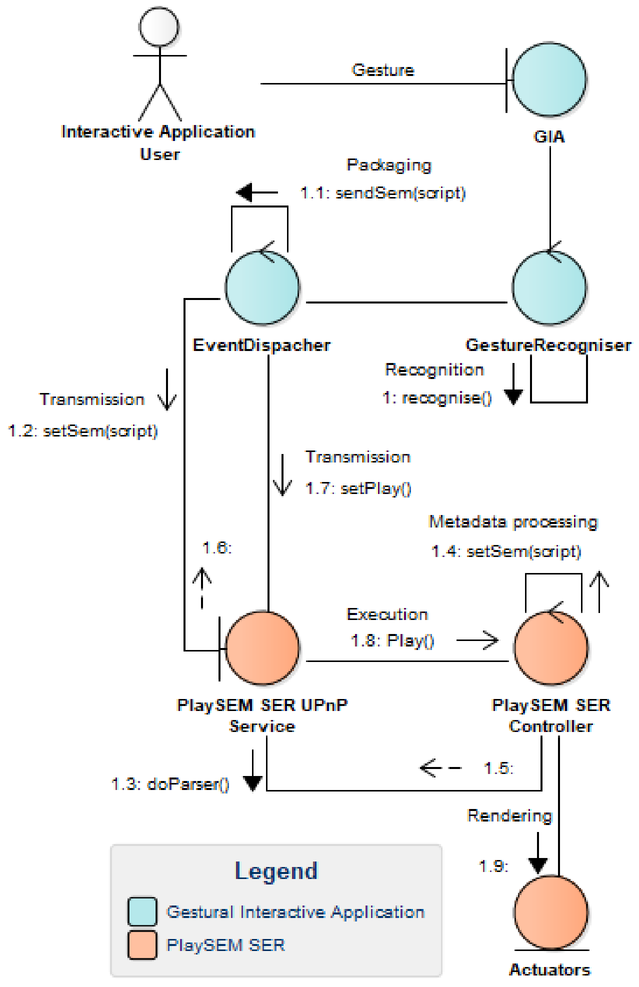

Figure 1: Original communication strategy adapted from [20]. Whenever a gesture occurs, this sequence is performed.

of unnecessary real-time communication between client applications and the PlaySEM SER, (ii) reducing the number of exchanged messages through the network, and (iii) adding lightweight communication protocols from the CPS domain in order to reduce response time.

Based on a feature of the PlaySEM platform that requires publish/subscribe messaging pattern to process asynchronous messages, we selected the protocols CoAP, MQTT, and WebSocket, apart from UPnP which is natively supported by the PlaySEM SER. Considering that UPnP puts together a set of protocols such as HTTP, SOAP and XML on top of IP, we aim at answering if other protocols can improve response time interval in networked event-based mulsemedia applications provided that they efficiently carry less information, thereby reducing the time for communication.

CoAP is a transfer protocol designed for use with constrained environments, that is, systems with restricted conditions [23]. It is primarily used in IoT and CPS which require lightweight protocols to communicate with limited devices and networks. CoAP is based upon Representational State Transfer (REST) architecture providing a request/response and publish/subscribe interaction with very low overhead. CoAP messages are exchanged asynchronously between CoAP endpoints. It runs on UDP, however, it has its own reliability mechanism when it is needed a message confirmation. According to [23], it could also be used over other transports such as SMS, TCP, or SCTP. In [12], the authors reported that their framework, entitled as Californium CoAP, showed 33 to 64 times higher throughput than high-performance HTTP Web servers. Thus, besides meeting the requirements of the PlaySEM platform, it has a promising performance in distributed systems.

Just like CoAP, MQTT is a lightweight protocol thought to work with resource-constrained devices [2]. However, it is limited to a publish/subscribe interaction based on topics and runs on top of TCP/IP. The protocol itself is designed to be simple. A client can publish messages to the broker and subscribe to topics for receiving events. The protocol has three qualities of service for message delivery. These are: *At most once*, with a low guarantee, in which message loss can occur; *At least once*, with guaranteed delivery but duplications can happen; and *Exactly once*, in which is assured that a message will arrive just once. The intrinsic characteristic of MQTT makes it a minimised protocol to reduce network overhead.

Unlike CoAP and MQTT, the Websocket protocol was not designed originally for constrained devices. It aims at making available a mechanism for browser-based applications, such as games, simultaneous editing tools, user interfaces exposing server-side services in real time, and so on, that need two-way communication with servers that do not rely on opening multiple HTTP connections [7]. In contrast to polling, it does not have to repeat HTTP headers in each request, attenuating communication overhead. It helps developers to build scalable real-time web applications through a single socket over the internet via browser [18]. The fact of many applications nowadays are implemented in browsers, and most of them provide support to Websockets, makes it a requirement for the PlaySEM SER to incorporate.

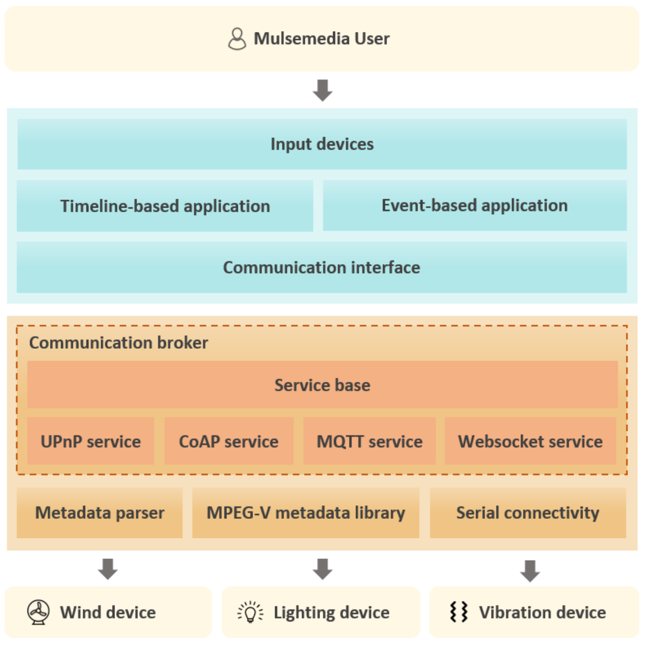

Figure 2: The new architecture of the PlaySEM platform.

Despite the fact that these protocols work differently, the services provided by the PlaySEM SER are the same. Figure 2 shows the new architecture of the PlaySEM platform highlighting the dashed rectangle called *Communication broker*. It allows client applications to choose the best interface for them not based on performance alone, but also on their requirements. For instance, manipulating Websockets on web applications requires less effort than UPnP. The new PlaySEM architecture makes an abstraction of the services and includes an interface for each protocol used for accessing these services. It also enables the PlaySEM SER to extend the range of supported protocols without changing its services.

In Table 1 the PlaySEM's components and its new architecture are presented. It is introduced in blocks representing important features of the whole platform. Notice that the *Mulsemedia User* and *Wind, Lighting, and Vibration devices* play a role in Figure 2 but they are not components of the architecture.

Table 1: Description of the new architecture of the PlaySEM platform and its main blocks.

Block	Description
Input devices	*When developing an event-based application, such as a game or an interactive application, events have to be captured. It represents any sensor such as an RGB-D camera or a joystick.*
Timeline-based application	*It is a kind of application guided by a timeline such as video or music players.*
Event-based application	*It is a kind of application whereby events captured from the physical world are conveyed to the virtual world and sparks an action.*
Communication interface	*It is a component responsible for communicating with external applications.*
Communication broker	*It is a mediator. It receives messages from the sender and translates them internally to call services.*
Service base	*It contains the implementation of each service exposed by the protocols.*
UPnP service	*It enables a UPnP interface for clients to consume the services.*
CoAP service	*It enables a CoAP interface for clients to consume the services.*
MQTT service	*It enables an MQTT interface for clients to consume the services.*
Websocket service	*It enables a Websocket interface for clients to consume the services.*
Metadata parser	*It is responsible for converting MPEG-V metadata scripts into commands for the devices.*
MPEG-V metadata library	*It is a library for marshalling and unmarshalling MPEG-V XML files into Java objects.*
Serial connectivity	*It supports serial connectivity with the devices.*

3.3 Communication strategy

In the experiment related in [20], the authors recommended avoiding the tasks of repackaging, retransmission and reprocessing MPEG-V metadata script in real-time to reduce the overhead and improve response time in the integration of the GIA and the PlaySEM SER. In their application, once a gesture is recognized, a new message with a metadata should be send to the PlaySEM SER, even for a gesture which has been recognized previously. It injects an unnecessary overhead whenever a gesture occurs, and some of the tasks depicted in Figure 1 can indeed be made more efficient.

To improve the communication strategy between the GIA, or any other event-based application, and the PlaySEM SER, a two-step process was created. The goal is to reduce repetitive tasks and the number of exchanged messages through the network. As an alternative to the services *SetSem* and *SetPlay* (see 3.1), two new services called *SetSemEvent* and *SetPlayEvent* were included in the original implementation of the PlaySEM SER to deal specifically with event-based applications. Fundamentally, they carry the parameter *eventId* to make distinction between the scripts.

In the first step, the GIA and the PlaySEM SER have to handshake soon after loading in order to transmit and process all scripts before the user starts interacting with the GIA. This process is shown in Figure 3.

After initialising, the GIA packages all the MPEG-V metadata scripts to be sent to the PlaySEM SER Communication Broker (*Packaging task*). Following that, it starts a loop in order to send and process each script. The message *SetSemEvent* containing an MPEG-V script related to the recognised gesture and an identification (*eventId*) is transmitted through the network (*Transmission task*). Once received, the PlaySEM SER Communication Broker calls the PlaySEM SER Controller to parser the script. Subsequently, the PlaySEM SER sends a confirmation message to the GIA to signal that the event was processed and can be executed (*Transmission task*).

In the second step, both applications behave almost as they did in the original strategy, however, without needing to repackage, retransmit and reprocess MPEG-V metadata script in real-time. This process is shown in Figure 4.

After capturing a gesture with the Kinect (RGB-D sensor), the GIA recognises it (*Recognition task*). Every gesture is related to one or more action, which is considered an event for the PlaySEM SER. Each of them is identified with an *eventId*. Thus, after recognising a gesture, the GIA just sends the message *SetPlayEvent* to indicate that the execution can start (*Transmission task*). Finally, the script is executed (*Execution task*), and then sensory effects are rendered in the user's environment (*Rendering task*). As can be seen, message traffic is automatically decreased, once there is no more need to perform as many tasks as depicted in Figure 1.

4 EXPERIMENTAL DESIGN AND SETUP

The new communication strategy previously described was used to measure the efficiency of the communication between the GIA and the PlaySEM SER in real-time. The main goal was to identify the effect on response time from the reduction of operations, and from

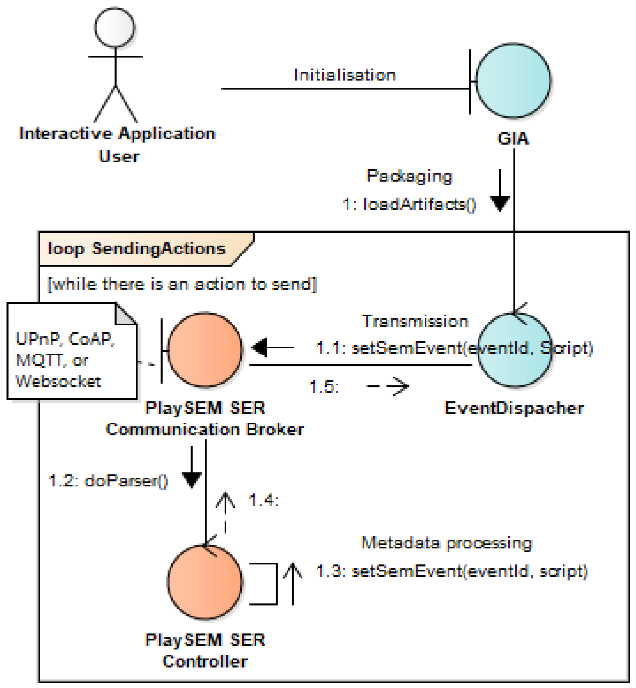
Interactive Application User
Initialisation
GIA
Packaging
1: loadArtifacts()
loop SendingActions
[while there is an action to send]
UPnP, CoAP, MQTT, or Websocket
Transmission
1.1: setSemEvent(eventId, Script)
1.5:
PlaySEM SER Communication Broker
EventDispacher
1.2: doParser()
1.4:
Metadata processing
1.3: setSemEvent(eventId, script)
PlaySEM SER Controller

Figure 3: Step one - modified communication strategy for loading MPEG-V scripts and pre-processing them.

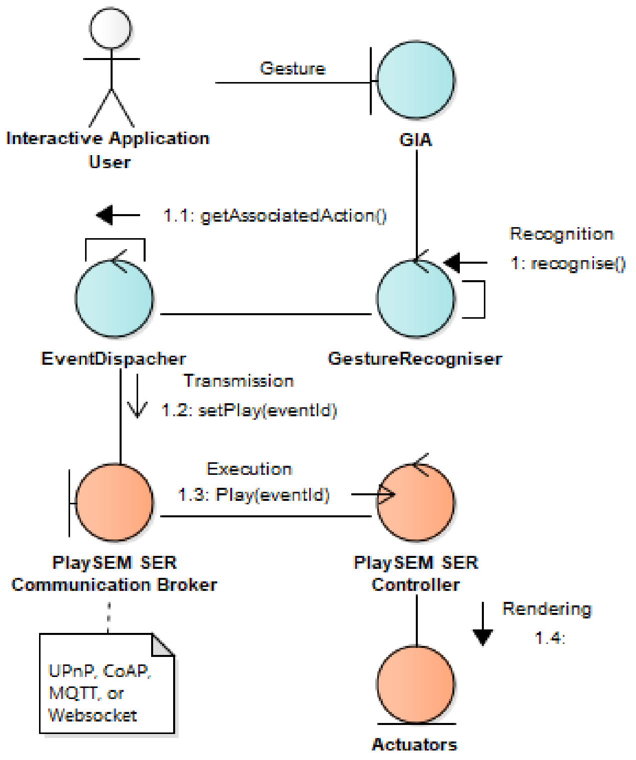

Figure 4: Step two - optimised strategy. New sequencing whenever a gesture occurs.

the protocols added to the PlaySEM SER. The performance of the protocols was measured with regard to time in which a message is transferred and confirmed in a wired network *Transmission task*.

4.1 Method

As reported in Section 3.3, there are two steps for the integration between the applications. In the first one, MPEG-V scripts associated to each user interaction are loaded in the PlaySEM SER and pre-processed when the GIA is loading. It is done just once. Thus, we are not interested in this response time because it could be done without the users realise it, for instance, at the time in which a splash screen is exhibited. Therefore, we focus on the second step, in which a real-time process is required and most of the critical operations are performed. Also, it can be directly compared to the previous approach described in [20].

The following tasks were performed in order to carry out the quantitative experiment.

(1) Computer *A* was set up to run mocks of the GIA whereas computer *B* configured to run the PlaySEM SER. The difference between the mocks and the GIA is that the former does not recognises gestures because we are not interested in improving recognition time in this paper.
(2) Computer *A* and *B* were connected to a router *R*, which was isolated from the internet and other devices.
(3) Fifty iterations of the message *SetPlayEvent* were triggered. Four mock applications (one for each protocol) were created to simulate the GIA keeping the same MPEG-V scripts (actions) as in [20]. Once there were 8 actions (*ACT02..ACT09*), the total number of iterations was 400. We did not distinguish them for the *SetPlayEvent* because the length is the same for each action and there was no transmission before it as in [20].
(4) The communication between the applications over the network was captured and its log stored in a text file.
(5) The set of data was converted into tabular sheets. For the CoAP protocol communication, the difference between the message *CON* sent by the client on the endpoint *SetPlayEvent* and the message *ACK* sent by the server was measured. For MQTT, the difference between *Publish Message* on the topic *SetPlayEvent* and *Publish Complete*. For UPnP, the difference between *POST* on the service *SetPlayEvent* and *HTTP/1.1 200 OK* was measured. Finally, for Websocket, the difference between *Websocket Text* containing a JSON (JavaScript Object Notation) object *SetPlayEvent* and [*ACK*] was also measured.
(6) To analyse the experimental results, we summarised the data set in measures of central tendency and variability.

4.2 Hardware and software setup

The hardware setup used in the experimental evaluation consists of two computers laptop and a router as mentioned in the previous Section. The first one, running the mocks for the GIA, is an *Intel Core i7-6700HQ, 16GB RAM, 254GB SSD*, running on Windows 10 64 bits. The second one, running the modified PlaySEM SER, is an *Intel Core i7-3537U, 4GB RAM, 128 GB SSD*, running on Windows 8.1 64 bits. They were connected via cable to the router *Wireless Dual Band AC750 Archer C20*.

For experiment, four applications simulating the GIA were created, one for each protocol. It was necessary to do it due to the way each protocol works. Nevertheless, the MPEG-V scripts were precisely the same. For COAP, there was a limitation when trying to send the script through the message *SetSemEvent* owing to its payload limitation. Thus, in that case, the PlaySEM SER received the scripts through Websocket and operated the service *SetPlayEvent* through CoAP. It is not a problem because the *SetSemEvent* time is not needed for the real-time conversation.

The PlaySEM SER was running under a simulated mode, which signifies that the output of the commands for playing sensory effects is exhibited on screen, instead of delivering it to the devices. For the modification of the PlaySEM SER to support the protocols CoAP, MQTT, and Websocket, open source implementations for Java such as *Californium CoAP framework 2.0.0-M4*, *Moquette 0.10*, and *Embedded Jetty 9.4.0.v20161208*, were respectively used. Also, we kept *Cling Core 2.0.1* for UPnP. CoAP ran on its *Confirmable Message* mode whereas MQTT on its *Exactly once*. For UPnP and Websocket the QoS level was set to default. A parameter was defined in the PlaySEM SER's configuration file to indicate which protocol would run in each instance.

Wireshark 2.4.2 was used to capture the communication between the mock and the PlaySEM SER applications over the network throughout the experiment. We present and discuss the acquired results in the next Section.

5 RESULTS AND DISCUSSION

Table 2 presents some measures of central tendency and variability for the service *SetPlayEvent* for each protocol. We calculated the mean, the standard error, the standard deviation, and the minimum and maximum value. Considering the mean, CoAP is the faster, followed by MQTT, UPnP, and Websocket. Regarding the standard error of the mean, the data set indicated little variability between sample means for each protocol. The standard deviation is higher for faster protocols. Since the mean is rather low for them, any data point far from the mean, or outliers, will affect the standard deviation severely. The minimum and maximum values agree on that, that is, they show how further the extreme data points are from the mean. Nevertheless, the data points are not spread out over a broader range of values as shown in Figure 5.

Figure 5 delineates the upper and lower boundaries for the outliers and does not indicate any pattern for them along the rounds. To the extent that the rounds proceeded, response time decreased slightly for UPnP. As a result, it could indicate that UPnP is faster as far as it goes, however, the frequency of the data points remained steady after the 200th round. Also, we can not conclude that UPnP is slower at the beginning of the rounds because the reason is unknown, and it was probably affected by extraneous variables. On the contrary, the frequency held relatively steady for CoAP, MQTT and Websocket.

5.1 Protocols timing and selection

Unlike timeline applications, in which some sensory effects can start before a particular scene, it is hard to know deterministically when an event will occur during the execution of an event-based one, such as a game, a VR/AR software, or any other interactive

Table 2: Central tendency and variability of the experiment's data set.

Protocol	Mean (ms)	SE Mean (ms)	StDev (ms)	Minimum (ms)	Maximum (ms)
CoAP	1.436	0.074	1.481	0.617	16.628
MQTT	3.288	0.087	1.740	1.937	33.251
UPnP	8.393	0.145	2.894	2.222	36.496
Websocket	51.210	0.121	2.430	41.162	74.037

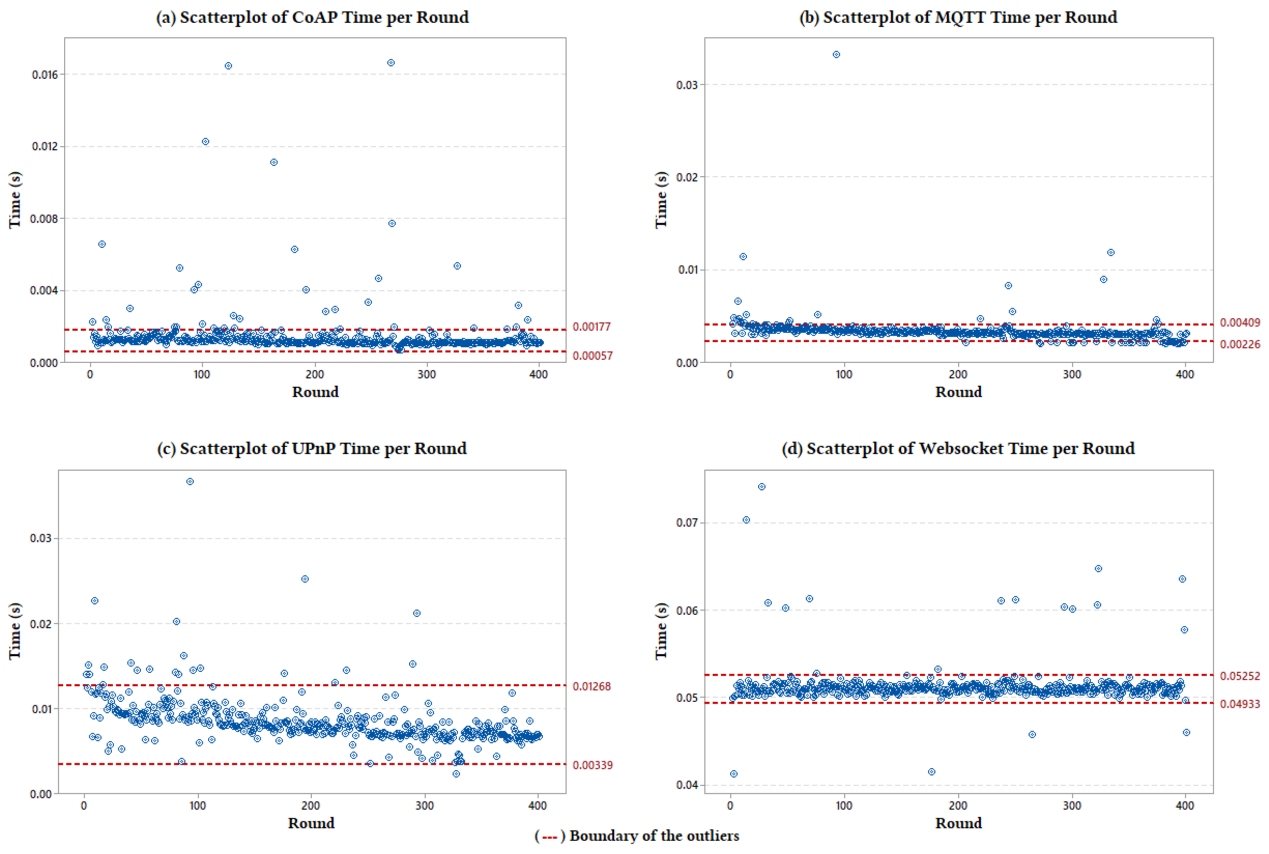

Figure 5: Dispersion of data for each protocol after 400 rounds.

application. Thus, synchronism must be tight, and response to an interaction should be given as soon as possible in order to avoid delays which might negatively impact users' expectation.

By adding lightweight communication protocols to the PlaySEM SER, we expected to cut down time wasting when integrating an event-based application to it via network. Figure 7 depicts the comparative mean between the protocols CoAP, MQTT, UPnP, and Websocket in the experiment. Since we created a new way of integration (see Figure 4) whereby MPEG-V scripts are pre-processed at the handshake of the applications (see Figure 3), it was needed to compare just the mean time from the service *SetPlayEvent*.

The results revealed that the selection of the protocol will have an influence on performance. Protocols commonly used in CPS such as CoAP and MQTT tend to deliver a response in a quite short time being the first one twice faster. Compared to CoAP, UPnP was 4 times slower, but it still remains as an option working around 8ms. Websocket hit 51ms on average. If on the one hand, it makes the process of integration easier for web applications, on the other hand, it could be infeasible for them to work efficiently. More research is needed with regard to the impact of delay in event-based mulsemedia systems for different types of sensory effects. In other words, this time could be reasonable for lighting but not for scent. However, the jury is still out on the issue of to what degree this kind of delay plus devices' delivery time will impact QoE of users.

In the event that developers devise their own communication protocol, they might enhance even more the response time interval in networked event-based mulsemedia systems. However, we are interested in scenarios where interoperability and reusability prevails and developers could take advantage of the PlaySEM SER to expand their multimedia applications to mulsemedia systems through industrial standards. Thereby, our conclusion is that developers are recommended to use CoAP for improving transmission

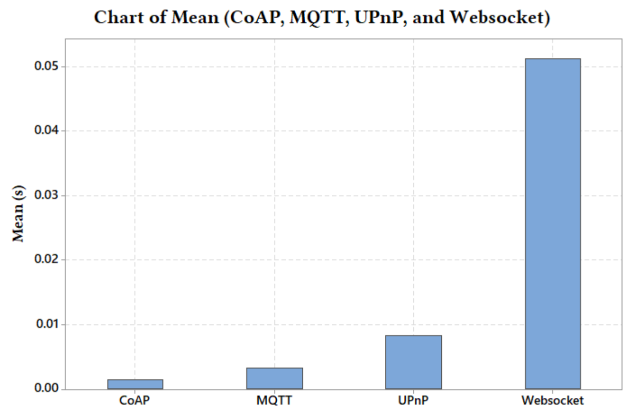

Figure 7: Comparative mean between the protocols CoAP, MQTT, UPnP, and Websocket in the experiment.

response time in similar cases when developing non-web applications. Despite the fact that Websockets are slower compared to the other protocols, we suggest its use when integrating mulsemedia web applications to the PlaySEM SER because the other protocols are not natively supported by web browsers and the total time in our integration's approach is equivalent of the total time pointed out in [20].

5.2 Communication strategy optimisation

Many tasks had to be performed in the original communication strategy reported in [20] from *Recognition task* until *Execution task* (see Section 3.1). By changing the communication strategy, we expected to reach a more efficient way of communication between the GIA and the PlaySEM SER, leading to a faster response. Though there has to be a loading step before the real-time conversation, it is not an issue. The upside is that pre-processing MPEG-V scripts prior to the required instantaneous communication will make the process slighter. The focus was on eliminating repetitive actions of packaging and sending scripts in real time and cutting down on the high number of exchanged messages.

Figure 6 compares the original to the current optimised communication strategy (see Section 3.3). The total time was defined by the sum ($\sum$) of each interaction step t_i represented by T for each protocol p starting from $i = 1$ until n. T_p represents the total time for the protocols CoAP (c), MQTT (m), UPnP (u), and Websocket (w). For the protocols, the range of time represented in t_{5p} was defined by the lower and upper outlier. Recognition time on the current strategy inherited values from [20] since we created mock applications to simulate the GIA.

As can be noticed in Figure 6, t_2, t_3, and t_4 were suppressed of the current strategy because the pre-processing of scripts was previously done when loading the applications. Therefore, we eliminated the time for the tasks *Packaging task*, the first *Transmission task*, and *Metadata processing task*. By doing so, the gain over the original strategy was between 19ms and 46ms. Furthermore, taking into account the time for the *Transmission task* obtained from the use of other protocols, t_5 decreased for the protocols CoAP and MQTT, held steady for UPnP, and increased significantly for Websocket. In spite of rising the latter, T_w was not severely affected by it in comparison with the original strategy due to the prior suppression of t_2, t_3, and t_4. Thereby, for T_c, T_m, and T_u there was a striking difference

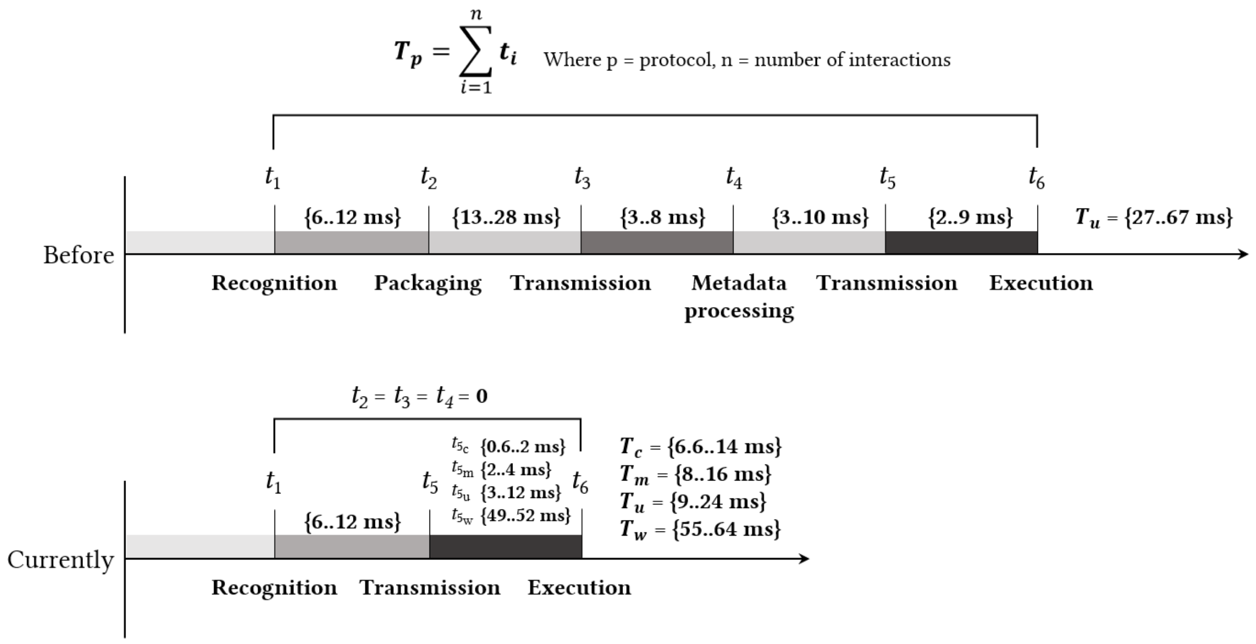

Figure 6: Time comparison between the original communication strategy reported in [20] and our approach whenever an event/gesture is detected.

to the original T_u strategy whereas T_w is considered equivalent in terms of superior threshold.

6 CONCLUSION

This work proposed a new way for improving response time interval in networked event-based mulsemedia systems from the change of the communication strategy between an event-based application and PlaySEM, an open source platform for rendering sensory effects over network. We contrasted the original communication strategy proposed by Saleme et al. [20] to our approach. In summary, the outcome demonstrated that pre-processing MPEG-V scripts for real-time communication in networked event-based mulsemedia systems could cut down time wasting significantly.

In addition, the use of lightweight communication protocols such as CoAP and MQTT boost response time up to four times compared to the native way of PlaySEM's communication based on UPnP. The worst case for CoAP, MQTT, and UPnP working over the new communication strategy outperforms the best case of the original strategy. All in all, the time for Websocket in the new strategy is virtually the same as the original strategy regarding superior threshold.

Despite employing the scenario of Saleme et al. [20], the results found through this research could be taken into consideration whenever working with any event-based mulsemedia application that describes its sensory effects in the standard MPEG-V and has a wish to make reuse of the PlaySEM platform for stimulating other senses beyond sight and hearing. Applications such as games, VR/AR software, and interactive applications fit well in this context.

Future work includes finding the influence of computational delays on the user's experience by identifying to what extent they are supported. Moreover, there is an open room for expanding the PlaySEM platform to make it even more flexible on the side of the devices by adding support to other connectivity protocols such as Ethernet 802.3, 802.11, Bluetooth, and Zigbee besides serial connectivity.

ACKNOWLEDGMENTS

Estêvão Bissoli Saleme thankfully acknowledges support from the Federal Institute of Espírito Santo. Prof. Celso A. S. Santos was supported by a research grant from CNPq (#312148/ 2014-3) and FAPES (#67927378/2015). Prof. Gheorghita Ghinea gratefully acknowledges funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 688503 for the NEWTON project (http://www.newtonproject.eu).

REFERENCES

[1] O. A. Ademoye and G. Ghinea. 2009. Synchronization of Olfaction-Enhanced Multimedia. *IEEE Transactions on Multimedia* 11, 3 (April 2009), 561–565. https://doi.org/10.1109/TMM.2009.2012927

[2] A. Banks and R. Gupta. 2014. MQTT Version 3.1.1, OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. (2014). Online; accessed 19 October 2017.

[3] S. Bartocci, S. Betti, G. Marcone, M. Tabacchiera, F. Zanuccoli, and A. Chiari. 2015. A novel multimedia-multisensorial 4D platform. In *2015 AEIT International Annual Conference (AEIT)*. 1–6. https://doi.org/10.1109/AEIT.2015.7415215

[4] H-Y. Cho. 2010. *Event-Based control of 4D effects using MPEG RoSE.* Master's thesis. School of Mechanical, Aerospace and Systems Engineering. Korea Advanced Institute of Science and Technology. Master's Thesis.

[5] M. Duchon, C. Schindhelm, and C. Niedermeier. 2011. Cyber Physical Multimedia Systems: A Pervasive Virtual Audio Community. In *MMEDIA 2011 - The Third International Conferences on Advances in Multimedia*. 87–90.

[6] R. Eg and K. Raaen. 2016. How to Demonstrate Delay?: Let's Play!. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 28, 4 pages. https://doi.org/10.1145/2910017.2910628

[7] I. Fette and A. Melnikov. 2011. The WebSocket Protocol. RFC 6455. https://tools.ietf.org/html/rfc6455. (2011). Online; accessed 19 October 2017.

[8] G. Ghinea, F. Andres, and S. R. Gulliver. 2011. *Multiple Sensorial Media Advances and Applications: New Developments in MulSeMedia: New Developments in MulSeMedia.* Information Science Reference.

[9] G. Ghinea, C. Timmerer, and W. Lin. 2014. Mulsemedia: State of the Art, Perspectives, and Challenges. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 1s, Article 17 (Oct. 2014), 23 pages. https://doi.org/10.1145/2617994

[10] S. Hoshino, Y. Ishibashi, N. Fukushima, and S. Sugawara. 2011. QoE assessment in olfactory and haptic media transmission: Influence of inter-stream synchronization error. In *IEEE Int. Workshop Tec. Committee on Communications Quality and Reliability (CQR)*. 1–6. https://doi.org/10.1109/CQR.2011.5996082

[11] S-K. Kim and Y. S. Joo. 2014. Sensible Media Simulation in an Automobile Application and Human Responses to Sensory Effects. *ETRI Journal* 35, 6 (Dec. 2014), 1001–1010. http://dx.doi.org/10.4218/etrij.13.2013.0038

[12] M. Kovatsch, M. Lanter, and Z. Shelby. 2014. Californium: Scalable cloud services for the Internet of Things with CoAP. In *2014 International Conference on the Internet of Things (IOT)*. 1–6. https://doi.org/10.1109/IOT.2014.7030106

[13] N. Murray, O. A. Ademoye, G. Ghinea, and G-M. Muntean. 2017. A Tutorial for Olfaction-Based Multisensorial Media Application Design and Evaluation. *ACM Comp. Surveys* 50, 5, Article 67 (2017), 30 pages. https://doi.org/10.1145/3108243

[14] N. Murray, B. Lee, Y. Qiao, and G-M. Muntean. 2014. Multiple-Scent Enhanced Multimedia Synchronization. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 1s, Article 12 (Oct. 2014), 28 pages. https://doi.org/10.1145/2637293

[15] N. Murray, B. Lee, Y. Qiao, and G-M. Muntean. 2017. The Impact of Scent Type on Olfaction-Enhanced Multimedia Quality of Experience. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47, 9 (Sept 2017), 2503–2515. https://doi.org/10.1109/TSMC.2016.2531654

[16] N. Murray, Y. Qiao, B. Lee, A. K. Karunakar, and G-M. Muntean. 2013. Subjective Evaluation of Olfactory and Visual Media Synchronization. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*. ACM, New York, NY, USA, 162–171. https://doi.org/10.1145/2483977.2483999

[17] National Science Foundation. 2017. Cyber-Physical Systems (CPS). https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286. (2017). Online; accessed 13 November 2017.

[18] V. Pimentel and B. G. Nickerson. 2012. Communicating and Displaying Real-Time Data with WebSocket. *IEEE Internet Computing* 16, 4 (July 2012), 45–53. https://doi.org/10.1109/MIC.2012.64

[19] M. Rank, Z. Shi, and S. Hirche. 2010. Perception of Delay in Haptic Telepresence Systems. *Presence: Teleoperators and Virtual Environments* 19, 5 (oct 2010), 389–399. https://doi.org/10.1162/pres_a_00021

[20] E. B. Saleme, J. R. Celestrini, and C. A. S. Santos. 2017. Time Evaluation for the Integration of a Gestural Interactive Application with a Distributed Mulsemedia Platform. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 308–314. https://doi.org/10.1145/3083187.3084013

[21] E. B. Saleme and C. A. S. Santos. 2015. PlaySEM: A Platform for Rendering MulSeMedia Compatible with MPEG-V. In *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web (WebMedia '15)*. ACM, New York, NY, USA, 145–148. https://doi.org/10.1145/2820426.2820450

[22] C. A. S. Santos, A. N. R. Neto, and E. B. Saleme. 2015. An Event Driven Approach for Integrating Multi-sensory Effects to Interactive Environments. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. 981–986. https://doi.org/10.1109/SMC.2015.178

[23] Z. Shelby, K. Hartke, and C. Bormann. 2014. The Constrained Application Protocol (CoAP). RFC 7252. https://tools.ietf.org/html/rfc7252. (2014). Online; accessed 19 October 2017.

[24] J. M. Silva, M. Orozco, J. Cha, A. E. Saddik, and E. M. Petriu. 2013. Human Perception of Haptic-to-video and Haptic-to-audio Skew in Multimedia Applications. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 2, Article 9 (May 2013), 16 pages. https://doi.org/10.1145/2457450.2457451

[25] I. M. L. C. Vogels. 2004. Detection of Temporal Delays in Visual-Haptic Interfaces. *Human Factors* 46, 1 (2004), 118–134. https://doi.org/10.1518/hfes.46.1.118.30394

[26] M. Waltl, B. Rainer, C. Timmerer, and H. Hellwagner. 2013. An End-to-end Tool Chain for Sensory Experience Based on MPEG-V. *Image Commun.* 28, 2 (Feb. 2013), 136–150. https://doi.org/10.1016/j.image.2012.10.009

[27] Z. Yuan, T. Bi, G-M. Muntean, and G. Ghinea. 2015. Perceived Synchronization of Mulsemedia Services. *IEEE Transactions on Multimedia* 17, 7 (July 2015), 957–966. https://doi.org/10.1109/TMM.2015.2431915