
Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTMENT OF COMPUTER ARCHITECTURE AND TECHNOLOGY

# INTEROPERABLE TECHNOLOGIES FOR MULTI-DEVICE MEDIA SERVICES

by:
Mikel Zorrilla Berasategi

Supervised by:
Dr. Alberto Lafuente Rojo
&
Dr. Julián Flórez Esnal

Donostia – San Sebastian, Wednesday 20th July, 2016


Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTMENT OF COMPUTER ARCHITECTURE AND TECHNOLOGY

# INTEROPERABLE TECHNOLOGIES FOR MULTI-DEVICE MEDIA SERVICES

by:
Mikel Zorrilla Berasategi

Supervised by:
Dr. Alberto Lafuente Rojo
&
Dr. Julián Flórez Esnal

Donostia – San Sebastian, Wednesday 20th July, 2016

This thesis is dedicated to my parents.

For their endless love, support and encouragement.

*Aita, ama, zuentzako.*

# Abstract

HTML5 is driving a strong trend towards interoperable Web-based applications, enabling a wider range of devices to run this kind of applications. However, most applications are running on these devices separated from each other or, at best are only loosely coupled. The growing interest in 2nd-screen solutions within the Connected TV sector clearly shows that users expect a more consistent experience across different devices and their applications. However, to do this, broadcasters and application developers currently need to implement, distribute and maintain a set of rather complex technical solutions tailored to each of the specific target platforms.

A more versatile solution would allow the implementation of applications independent from the target devices and the application itself would be able to run across multiple user devices. The user could then smoothly move parts of the functionality from one device to another in an intuitive manner and the application would adapt itself to the device. Essentially, the challenge is to take connected service development to a new level.

This research proposes interoperable technologies for multi-device media services in order to deal with the different involved challenges: multi-device adaptation, enabling broadcasters and developers to create a single application code, that will seamlessly adapted to a dynamic context of a user dealing with multiple devices at the same time; cross device synchronisation, enabling a shared state between the devices to exchange context and timing information and provide a synchronised experience across devices; and multi-connection, providing mechanisms to discover and associate different devices and users.

This research also proposes solutions to complement computing resources in both directions. On the one hand, a hybrid local-remote

rendering on client devices complemented with a cloud service is considered, when the media experience requires in real-time higher computing capabilities than the resources available in the client. This could be the case of computer generated 3D content. On the other hand, a solution with idle client devices as an infrastructure is proposed to save delay-tolerant computing workload on the cloud infrastructure. Thin devices could perform atomic tasks such as image analysis managed by the cloud service, to improve the experience of a social media service.

# Acknowledgements

First of all, I would like to thank all my colleagues in Vicomtech-IK4. More specifically those related to the MediaScape project. Igor G. Olaizola, thank you for making possible our participation in such a project and for your valuable contribution on its coordination; Iñigo Tamayo, thank you for your great support in the technical design and your incomparable skills for implementation; Ana Dominguez, thank you for your excellent attitude and aptitude addressing problems; Angel Martin, thank you for sharing with me all the research process and the literature writing. Esther Novo, thank you for supporting me and covering me during the whole MediaScape project. It is a pleasure to work with all of you.

I would also like to thank to my supervisors Julián and Alberto. I learnt a lot with you. Even more than expected at the beginning of the process. I hope that this will be reflected in the way I research in the future.

Part of the research of this Ph.D. has been done within the MediaScape European research project. The MediaScape project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement 610404.

Finally, I would like to express my gratitude to Vicomtech-IK4 for providing me a great environment to research and create this Ph.D. dissertation. Thank you to the directors Julián Flórez, Jorge Posada and Edurne Loyarte, and to the head of my department Igor G. Olaizola for trusting me since the first moment.

*Eskerrik asko guztioi*

Mikel Zorrilla Berasategi

July 2016

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction

CHAPTER

# 1

# Scope of the research

## 1.1 Motivation

We are currently witnessing a strong tide to powerful Web-based applications. This trend is also driving the evolution of HTML5, making a wider range of devices capable of running applications that gain features previously available only through native applications [Ant12].

Once the interoperability is being addressed by HTML5, the key challenge for next generation applications is to federate cooperative devices to provide users coherent multi-device experiences. This is a natural step in the adaptation of the market and society to the growing behaviour of users accessing services from several devices simultaneously [CD12]. However, most of the currently existing applications are running on devices independently or at best loosely coupled, sharing a back-end on an event-driven model. Therefore, a gap on the experience of users is produced, as they perceive devices as isolated pieces of applications when they would like to have a single experience through multiple devices at the same time.

Despite the continuous proliferation of smart devices, the TV set is still the prominent device to consume media services at home. Nonetheless, televisions are increasingly being combined with tablets, smartphones or laptops, all connected to the Internet. The growing interest in second screen solutions within the Connected TV industry [Cla12] clearly shows that users expect a more consistent experience across different devices and their applications [CBJ08] [Mil14].

However, not only the habits of the users while consuming media have changed. The media content and the associated services themselves have also suffered a great evolution. The continuous appearance of new technologies for computer generated 3D or immersive applications, allows creating attractive experiences by combining rich media. In this regard, 3D Media [ZDLB08] [DA09] refers to the media concept that incorporates the traditional audio and video sources with other new types such as computer generated 3D objects. Moreover, a media service could be composed by multiple media sources, such a multi-angle view, related information, interactive graphics or images.

In order to provide multi-device applications, broadcasters and media application developers currently need to implement, distribute and maintain a set of rather complex technical solutions tailored to each of the specific target platforms. The development of a second screen application including a TV and a mobile device is typically addressed by an event-driven approach [Zie13]. This implies broadcasters and application developers to create two different applications, one based on HbbTV [Mer11] on the TV and an HTML-based application for the mobile, as well as a server to handle the communication between applications. The need to implement customised applications for each platform, and the necessity to consider a coherent multi-device visualisation during the development phase, induces the development itself, test and maintenance of service-related applications to be inherently complicated.

Current solutions illustrate a fragmented market with vendor-specific, and often application-specific approaches. For Connected TV service provision (HbbTV, YouView, Android based platforms, proprietary technologies such as Samsung, LG, Philips, Sony, etc.) completely different developments have to be created and maintained in order to be compatible with all approaches and brands. So far, these multi-device applications are mainly restricted to a specific TV set and a second screen device (e.g. an Apple TV with an iPhone or iPad, or a Samsung TV with Samsung mobiles) or are fully disconnected, such as using Twitter hashtags.

The device fragmentation creates a wide spectrum of performance capabilities on the clients to consume media services. Powerful devices such as PCs and laptops, coexist with increasingly powerful but still thin ones such as TVs, set-top boxes and mobile devices. This enables a two-way complement for computing capabilities enhancement, driven by the favourable and increasingly improved communication conditions following the trend of 5G networks. On the one hand, a cloud service can complement in real-time thin devices when they require

higher computing capabilities, such as rendering computer generated personalised and interactive 3D content as part of the media service. On the other hand, thin devices can also complement a cloud server following the distributed computing paradigm for delay-tolerant work, since client devices are often idle or underperformed while watching videos and can contribute with part of their resources to perform atomic tasks of image analysis to enhance the media experience, such as creating automatic tagging, and save costs in the could resources.

Despite the clear opportunities to catalyse new business models, the lack of standards and technologies prevents the creation of seamlessly connected, intuitively converged and conveniently continuous experiences across a heterogeneous ecosystem of devices. A more versatile solution would allow the implementation of applications independent from the target devices and the application itself would be able to run across multiple user devices enabling the service pervasivity. The user could then smoothly move parts of the functionality from one device to another in an intuitive manner and the application would adapt itself to the device.

Essentially, the challenge is to take connected media service development to a new level, providing technology to create advanced connected multi-device and multi-user media services via a standardised approach integrated into the HTML5 paradigm. For this purpose, the technology has to facilitate the marriage of the TV, PC and Mobile worlds through a standard solution that includes real-time delivery and synchronisation of media contents and applications/services across a variety of devices being used simultaneously. Desirably, these new media services will imply computing resources capabilities both in the service provider's cloud infrastructure and in the end-user's client devices.

### 1.1.1 The Hybrid TV Ecosystem

In order to fully understand the motivation of this research work and to lay the foundations of the research objectives, this section provides pivotal conclusions, extracted from market surveys and reports, to be the base for the definition of a general overview of the Hybrid TV and multi-device experiences.

Hundreds of apps in the various market places aid consumers in controlling the first screen, discovering new content, enhancing their viewing experience and enabling a shared, social experience. These Hybrid TV and multi-device applications are completely oriented to a target device and lack a real interoperability between devices. As mentioned before and addressed in the following section,

there are limitations on the standards and technologies in the current state of the art to allow broadcasters and Web application developers to create this kind of application with universal compatibility in a sustainable way.

Due to the success of the Smart TV market there is an increasing demand of multi-device applications. As many HCI (Human Computer Interaction) researchers increasingly point out, Mark Weiser's vision of ubiquitous computing [Wei91] will be applied based on multiple devices: TVs and laptops, tablets and smartphones, watches and bracelets, etc. The real power of the concept comes from the interaction between devices and sharing experiences with other users.

To fully understand the scope of our research and its challenges, the following list summarises the considered contextual factors regarding the trend of media consumption habits of users together with market trends. This is reinforced and aligned with the general overview foreseen by [Cla12] and the accumulated experience from first level broadcasters and technology providers, such us BBC (British Broadcasting Corporation), BR (Bayerischer Rundfunk), IRT (Institut für Rundfunktechnik), NEC Europe (Nippon Electric Company), Web standardisation bodies like W3C (World Wide Web Consortium) and research institutes like Norut and Vicomtech-IK4 (Visual Interaction & Communication Technologies) [Med14]:

1. Broadcast capable devices are highly relevant for media applications. TV and radio are still the most common devices to consume media services. In 2014 TV dominates the daily media consumption, where most of the content was delivered to the TV set in a traditional manner: over terrestrial, cable or satellite broadcast, as well as IP delivery networks [Mil14] [Nie14a]. At the same time, radio reaches all people age 12+ during the week [Nie14b]. Radio and TV are in a good position to capture the attention of the audience and drive their interests to specific contents or services.

2. Broadcast capable devices are increasingly connected to Internet. Global Smart TV shipments grew 55% year-over-year where around 50 percent of Smart TV owners across the USA and major European markets are currently using their TV's Internet capabilities [Str14]. Internet radio devices are still immature, evolving to be able to receive and play streamed media from internet radio stations and contents from the home network but often lack a real integration of broadband and broadcast services.

3. Ubiquitous computing centred in the home is a relevant scenario for multi-device media services for end users, broadcasters and application developers. It also brings challenges of ubiquitous computing related to mobility of the users through different contexts and networks, different rooms, multi-member households, going out from home, continuing the experience when arriving at home, etc.

4. Users have other connected devices while watching TV in a home scenario. The consumption habits have changed completely with many TV viewers using tablets, smartphones or laptops while watching TV. According to a study of Ericsson ConsumerLab [An 13] social networking sites and forums are used by 59% of people while watching TV, 49% will use apps or browse the Internet to find out more about the content they are watching, while almost 1 in 3 will discuss what they are currently viewing over social networks or chats.

5. Users are often using two devices simultaneously, accessing to related information in one device to complete the content being watched in the other. Fuelled by the deep penetration rate of smartphones, tablets and laptops worldwide [Gar14], 35% of first screen time is second screened, of which 1/3 is related to the main content [Mil14]. Concerning second screen companion experiences, 63% of the consumers accessing synchronised content on the second screen say it makes them feel a better experience in the shows they are watching [NAP14b].

6. Social experiences relate to a media application. The relation of broadcasted contents and social networks is two way. On the one hand, broadcasted contents are the main driver of social activity at prime time engaging audience to share immediate reactions [Nie14c]. On the other hand, the social media has the ability to engage TV audiences in real-time and in the days that follow. Regarding second screen companion experiences, 14% of second screen time is related to social activity about TV program [Nie14d], chat about a live broadcasted programme, suggesting content, share impressions, etc.

Once the focus of the work has been stated, Figure 1.1 presents a diagram of the general overview of the hybrid TV and multi-device experiences. On the one hand, the TV and the radio devices have the broadcast capability to access media

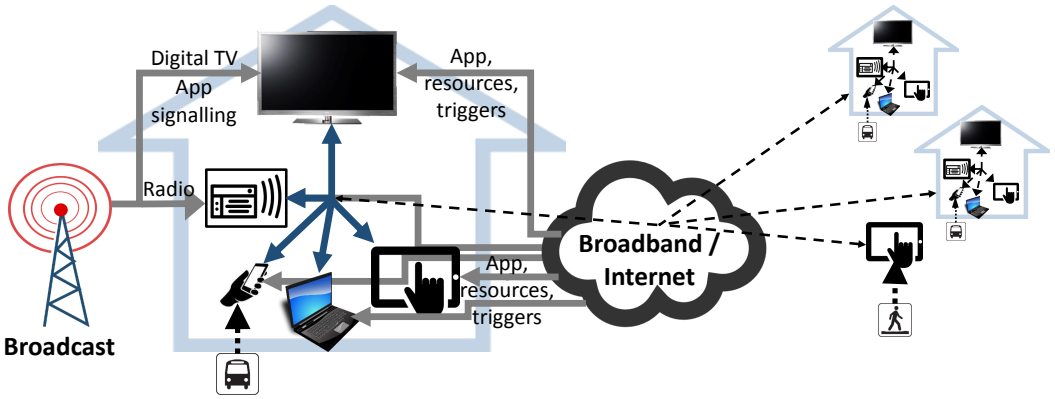


**Figure 1.1:** Diagram of the general overview of the Hybrid TV and multi-device experiences

content. Moreover, broadcast-related specifications, such as HbbTV, allow the hybrid application signalling through the broadcast channel. On the other hand, the media devices at home, such as tablets, smartphones and laptops, are connected to the Internet. This includes the connected TVs and radios that have the hybrid broadcast-Internet capability creating a bridge between the two channels. Internet brings a way to access applications, content and delivery of application triggers. Accordingly, Figure 1.1 shows all the media devices at home interconnected between them, offering the user a single experience through multiple displays. On the same way, the figure represents the home-centric *on the move* scenarios, where a user arrives at home with a smartphone and the application adapts to the home context. Finally, the icons on the right side show other homes and users *on the move*, sharing a multi-user experience.

## 1.2 Hypothesis

The working hypothesis is constructed as a statement of the following expectations:

1. Broadcasters and/or application developers will create a single application that will automatically adapt to a dynamic multi-device environment

2. End-user will perceive a single experience through multiple devices. In other words:

   (a) End-users will be able to migrate the application from one device to another in a consistent way (see Figure 1.2a)

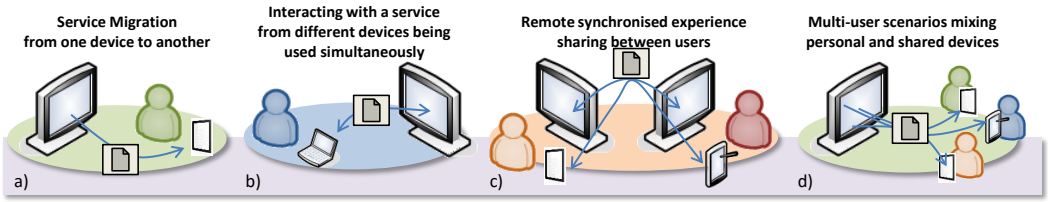


**Figure 1.2:** Diagram of the hypothesis of the perception of the users for a single experience through multiple devices

(b) End-users will be able to run the service across different devices at the same time splitting the functionalities of the application through the different devices (see Figure 1.2b)

(c) End-users will be able to share a synchronised experience with a remote user or a community (see Figure 1.2c)

(d) End-users will be able to consume the service mixing shared devices, such as a TV or a big screen, and personal devices, such as mobiles or laptops (see Figure 1.2d)

3. There will be a two-way complementarity of the computing resources in the multi-device media services:

(a) When client-devices are not powerful enough to render complex media services, the service will enable a hybrid local-remote rendering to provide the client-device the required real-time computing capabilities

(b) When service providers require highly-demanding but delay-tolerant computing resources to improve their services, such as generating automatic tagging of the content, the service will expose the client-devices as an infrastructure to distribute the processing tasks among all clients and reduce the workload in service provider's cloud server

The expectations of the working hypothesis involve different stakeholders:

1. the brodcasters and application developers, facilitating the creation of this kind of application without dealing with a set of complex technologies;

2. the end-users, being able to consume a multi-device media service as a single experience through multiple devices and users at the same time;

3. the service provider, being able to provide a two-way complement of computing resources.

The following section describes the objectives of the research, as a verification of the stated expectations in the working hypothesis.

## 1.3 Objectives

The main objective of this work is to identity and present a versatile and interoperable solution to allow the implementation of multi-device media services, independently from a specific target device or platform. Thus, it is necessary to overcome the developing paradigm of multi-device media services:

- Lay the basis to extend HTML-based applications towards a new standard and interoperable way to specify and develop advanced connected media services
- Provide technologies to enable the adaptation of the application to the user context in ubiquitous and simultaneous multi-screen scenarios, and in synchronised shared experiences between users or communities

To achieve the main objective, it is necessary to address and provide solutions to overcome the three main challenges of multi-device media services (see Figure 1.3):

- Multi-device adaptation: The application itself will adapt to a dynamic multi-device context of the end-user, providing a single experience through multiple devices at the same time
- Cross device synchronisation: All the content will be perfectly synchronised across different devices, taking into account all the media sources, as well as the associated data.
- Multi-connection: The application itself will discover the services and other available devices, associating multiple devices to a single user and multiple users to a shared experience, as well as providing mechanisms for a cross platform authentication.

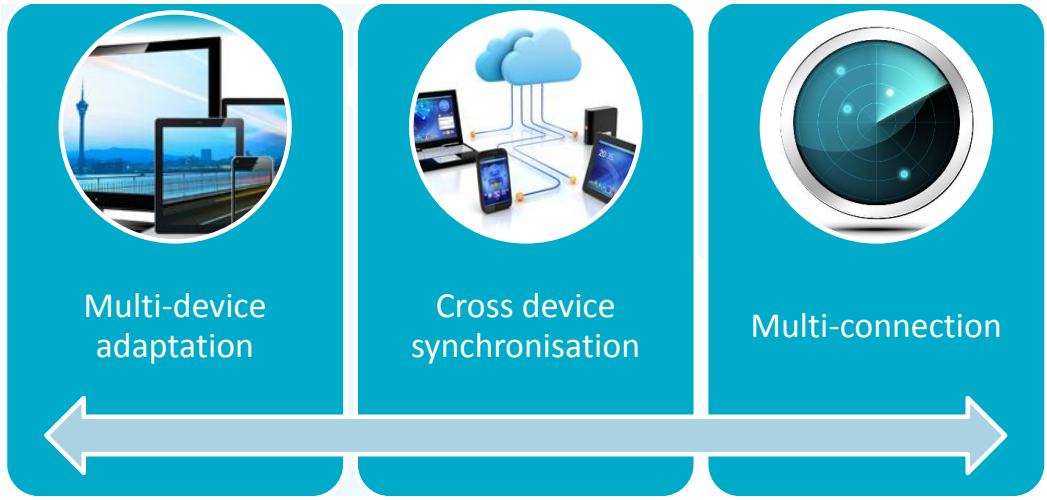


**Figure 1.3:** Main challenges to address to achieve the main objective

In order to provide technologies to create media services over multiple and fragmented devices, a new objective also has been defined to complement the computing requirements in both directions:

- A cloud server computing to complement thin devices
- Thin devices as an infrastructure to complement a cloud server

## 1.4 Methodology

The research work performed in this Ph.D. has followed a methodology based on the design cycle (see Figure 1.4). Based on a motivation for a research topic, a working **hypothesis** has been constructed as a statement of expectations. While the outcome of the research are the **contributions**, to achieve them an iterative process has been followed:

- **Design**: First of all, the design objectives have been clearly defined to be addressed during the architectural design phase of the different aspects of the solution.
- **Implementation**: Proof-of-concept implementations have been performed following the architectural design.
- **Evaluation**: The design objectives have been verified by terms of testing the proof-of-concept implementation.

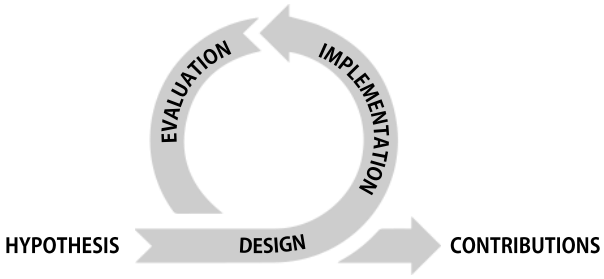


**Figure 1.4:** Diagram of the design cycle followed as the methdology of research

The iterative process has helped to refine the design and the implementation of the solution every iteration, resulting in the final contributions of the research.

## 1.5 Contributions

The main contribution of this Ph.D. research has been the creation of interoperable technologies to provide multi-device media services, laying the foundations to extend HTML-based applications, such as HbbTV or HTML5, towards standard solutions that enable fully manageable, context-sensitive, controllable and self-regulated multi-device applications.

More specifically, the main contribution can be translated to five specific outcomes:

1. Identify, design and implement a more versatile and interoperable solution to allow the implementation of multi-device media services, with the creation of a single application code that will adapt automatically to a dynamic user multi-device configuration across heterogeneous devices, providing an adaptive user interface on each device and a single experience across multiple devices at the same time.

2. Identify, design and implement a solution to manage the maintenance of a session in the cloud service to enable a shared state between different devices for sharing contextual and timing information. This solution allows to provide synchronisation across multiple devices.

3. Identify, design and implement a solution to discover devices around an HbbTV television, associating the TV and the discovered devices to the same user, using local network or physical proximity mechanisms.

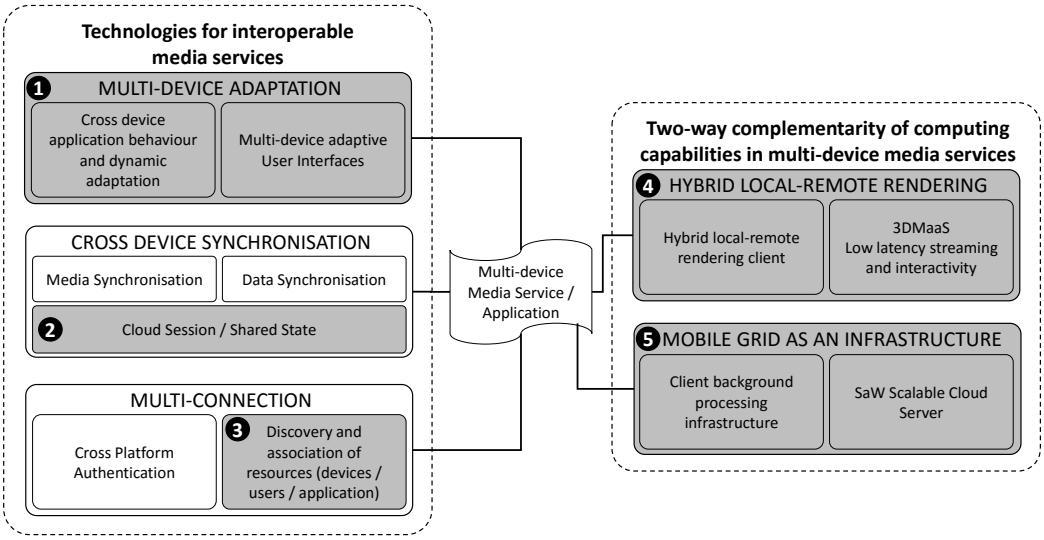


**Figure 1.5:** Diagram of the contributions of the research. The numbers reflect the five specific contributions of the research.

4. Design and implement a proof-of-concept solution for a hybrid local-remote rendering to enhance the real-time computing requirements of the client devices with cloud resources.

5. Design and implement a proof-of-concept solution to complement a cloud infrastructure for delay-tolerant computing requirements with a set of connected thin devices, saving costs to the service provider at the server side.

Figure 1.5 illustrates the five contributions of the research in a wider context to address the creation, delivery and management challenges of multi-device media services.

The contributions produced as an outcome the following publications:

- Publications regarding Contribution 1:
  - *Zorrilla, M., Borch, N., Daoust, F., Erk, A., Flórez, J., & Lafuente, A. (2015). A Web-based distributed architecture for multi-device adaptation in media applications. Personal and Ubiquitous Computing, 19(5-6), 803-820.* in Section 2.2
  - *Zorrilla, M., Tamayo, I., Martin, A., & Dominguez, A. (2015, June). User interface adaptation for multi-device Web-based media applications. In 2015 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (pp. 1-7). IEEE.* in Section 2.3

- Publications regarding Contribution 2:
  - *Zorrilla, M., Tamayo, I., Martin, A., & Olaizola, I. G. (2013, June). Cloud session maintenance to synchronise hbbtv applications and home network devices. In 2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) (pp. 1-6). IEEE.* in Section 2.4
- Publications regarding Contribution 3:
  - *Zorrilla, M., Martin, A., Tamayo, I., O'Halpin, S., & Hazael-Massieux, D. (2014, June). Reaching devices around an HbbTV television. In 2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (pp. 1-7). IEEE.* in Section 2.5
- Publications regarding Contribution 4:
  - *Zorrilla, M., Martin, A., Sanchez, J. R., Tamayo, I., & Olaizola, I. G. (2014). HTML5-based system for interoperable 3D digital home applications. Multimedia Tools and Applications, 71(2), 533-553.* in Section 3.2
- Publications regarding Contribution 5:
  - *Zorrilla, M., Martin, A., Tamayo, I., Aginako, N., & Olaizola, I. G. (2013, September). Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis. In Cloud and Green Computing (CGC), 2013 Third International Conference on (pp. 389-396). IEEE.* in Section 3.3
  - *Zorrilla, M., Flórez, J., Lafuente, A., Martin, A., Olaizola, I. G., & Tamayo, I. (Submitted in 2016, July). SaW: Video Analysis in Social Media with Web-based Mobile Grid Computing. IEEE Transactions on Mobile Computing* in Section 3.4

## 1.6 Document structure

This document contains three main parts. In Part I an introduction to the research scope is presented, focusing in the motivation for the research, the main objectives, the hypothesis, the methodology and the main contributions of the Ph.D. work. In Part II the research results are described having two main chapters:

- In Chapter 2 the contributions to create multi-device media services are described by providing interoperable technologies to overcome the main challenges of multi-device adaptation (Contribution 1), cross device synchronisation (Contribution 2) and multi-connection (Contribution 3).

- In Chapter 3 the contributions to create a solution that enables a two-way complementarity of the computing resources for multi-device media services are described (Contributions 4 and 5).

In Part III the main conclusions of the research can be found, including a discussion that enables future work.

Finally, IV provides other publications of the author and his Curriculum Vitae as an appendix, while V contains the bibliography.

# Part II

# Research Results

CHAPTER

# 2

# Technologies for interoperable multi-device media services

## 2.1 Context

In order to gain leverage from ubiquitous computing to multi-device applications, it is essential to advance in different areas that span both application developer requirements and user expectations. This research has focused on three main challenges:

1. **Multi-device adaptation** (Contribution 1), spanning autonomous creation of adaptive interfaces suitable to each concurrent viewer, and automatic interface arrangements tracking dynamic application logic;

2. **Cross device synchronisation** (Contribution 2), providing hybrid broadcast-Internet media synchronisation and deploying a shared context available immediately to all participants to provide session awareness and shared timing information for media synchronisation; and

3. **Multi-connection** (Contribution 3), addressing the discovery and association of the available resources, such as users, devices and services.

Facing the **multi-device adaptation** challenge means to evolve from developing different applications for specific target devices —controlling one by one their functionality, their interfaces and creating specific event-driven mechanisms for inter-device communication—, to a single application —where developers describe the complete functionality, interface and the behaviour of the application for a multi-device environment once—. The seamless translation of a single application into a multi-device execution while still providing a well fitted portion of the application on each device, is the multi-device adaptation challenge addressed in Section 2.2.

The main contribution of the research in multi-device adaptation is the design and implementation of a distributed adaptation architecture for media-driven multi-device applications. A distributed architecture is provided where the application developers create a Web application in terms of logic components once and add properties to define the multi-device behaviour of the application on a high abstraction level. Thus, during the development phase, the application developers do not need to be aware of a particular multi-device context or define views for a specific set of devices (TVs, radios, mobiles, laptops, etc.). Developers have to define global hints to allow the architecture decide the best configuration for visualisation through a dynamic set of involved devices. The libraries and tools comprised in the solution are responsible for analysing and managing the multi-device context of the application and the properties defined in the application to:

(a) decide which logic components are presented on each device on a specific moment,

(b) create a responsive layout, seamlessly adapted to each device, and

(c) generate a seamless inter-device communication channel.

The user would then be able to smoothly move parts of the application from one device to another in an intuitive manner, and the application would self-adapt to each device.

Section 2.3 complements the research in multi-device adaptation, presenting user tests performed over 47 end users to analyse the impact of different parameters on how to arrange the User Interface for simultaneous Web-based multi-device media applications. The conclusions extracted from the user testing activities help application developers to design a unique multi-device media

application that will provide a coherent experience across multiple devices at the same time, with a responsive User Interface on each device depending on the parts of the application that is being presented regarding the multi-device dimension context.

Regarding **cross device synchronisation**, Section 2.4 presents a solution to synchronise a multi-device execution, using a shared state as a cloud session in the server side, with information about timing and the multi-device context. The solution enables the incorporation of HbbTV applications (both v.1.1 and v.1.5) as part of the multi-device execution, together with mobile devices (e.g. Android or iOS smartphones and tablets) through standard Web technologies. Thereby the server keeps and manages session variables, available intermediately to all participants, with context and timing information among others.

Addressing **multi-connection**, Section 2.5 focuses on the resource discovery and association procedures in order to be able to provide multi-screen media services around an HbbTV television. It describes a the discovery and association challenges in distributed and pervasive computing, translating those challenges to the home environment around a TV. In the research work different solutions are proposed as contributions to discover and associate devices in a multi-device media service:

(a) Visual solutions like QR codes,

(b) sound patterns for devices that are physically closed, or

(c) a bump-based solution based on an event-driven server that receives requests from different devices and associates them according to the *timestamp* of the request and some additional information of the request, such as the geolocation.

Moreover, as an outcome of the research within the multi-connection challenge, a three-step solution has been proposed as a proposal combining the aforementioned methods for device discovery and association.

## 2.2 A Web-based distributed architecture for multi-device adaptation in media applications

- **Title:** A Web-based distributed architecture for multi-device adaptation in media applications

- **Authors:** Mikel Zorrilla, Njål Borch, François Daoust, Alexander Erk, Julián Flórez, Alberto Lafuente
- **Journal:** Personal and Ubiquitous Computing
- **Publisher:** Springer
- **Year:** 2015
- **DOI:** http://dx.doi.org/10.1007/s00779-015-0864-x


**Abstract.** HTML5 is driving a strong trend towards interoperable Web-based applications, enabling a wider range of devices to run this kind of applications. The key challenge of next generation media applications is to federate cooperative devices to provide multi-device experiences overcoming current second screen solutions within the Connected TV industry. There is a gap on the experience of users, since they perceive devices as isolated pieces of applications when they would prefer to have a single experience through multiple devices at the same time. This paper proposes a unified methodology and a common specification over Web Components for the adaptation of a single application, seamlessly running different instances on one or more devices simultaneously, according to the multi-device context of the user and the specific features of the devices. The solution presented in this paper extends current Web standards towards an interoperable architecture, and offers broadcasters and media application developers the possibility to easily design applications that will automatically provide a unique consistent experience across the connected devices. The architectural design is targeted to be included in the roadmap of the standards.

**Keywords:** Multi-device adaptation, responsive adaptive user interface, ubiquitous computing, pervasive computing, media-driven Web application, broadcasting

### 2.2.1 Introduction

We are currently witnessing a strong tide to powerful Web-based applications. This trend is also driving the evolution of HTML5, making a wider range of devices capable of running applications that gain features previously available only through native SDKs [Ant12].

Once the self-capacity and interoperability are being addressed by HTML5, the key challenge for next generation applications is to federate cooperative devices to provide users coherent multi-device experiences. This is a natural step in

the adaption of the market and society to the growing behaviour of users accessing services from several devices simultaneously [CD12]. However, most of the currently existing applications are running on devices independently or at best loosely coupled, sharing a backend on an event-driven model. Therefore, a gap on the experience of the users is produced, as they perceive devices as isolated pieces of applications when they could have a single experience through multiple devices at the same time.

Despite the continuous proliferation of smart devices, the TV set is still the prominent device to consume media services. Nonetheless, televisions are increasingly being combined with tablets, smartphones or laptops, all connected to the Internet. The growing interest in second screen solutions within the Connected TV industry [Cla12] clearly shows that users expect a more consistent experience across different devices and their applications [CBJ08] [Pes].

In order to provide multi-device applications, broadcasters and media application developers currently need to implement, distribute and maintain a set of rather complex technical solutions tailored to each of the specific target platforms. For a second screen application including a TV and a mobile device, [ZTMO13] and [Zie13] present an event-driven approach. They propose a solution with two different applications, one based on HbbTV [Mer11] on the TV and an HTML-based application for the mobile, as well as a server to handle the communication between applications. The need to implement customised applications for each platform, and the necessity to consider a coherent multi-device visualisation during the development phase, induces the development itself, test and maintenance of service-related applications to be inherently complicated.

Current solutions illustrate a fragmented market with vendor-specific, and often application-specific approaches. For Connected TV service provision (HbbTV, YouView, Android based platforms, proprietary technologies such as Samsung, LG, Philips, Sony, etc.) completely different developments have to be created and maintained in order to be compatible with all approaches and brands. So far, these multi-device applications are mainly restricted to a specific TV set and a second screen device (e.g. an Apple TV with an iPhone or iPad, or a Samsung TV with Samsung mobiles) or are fully disconnected, such as using Twitter hashtags.

Despite the clear opportunities to catalyse new business models, the lack of standards prevents the creation of seamlessly connected, intuitively converged and conveniently continuous experiences across a heterogeneous ecosystem of devices. The priority for future research is shifting towards fully manageable,

context-sensitive and controllable or self-regulating multi-device applications. Moreover, considering the distributed nature of the devices that access the services, some additional capabilities are necessary, such as autonomous information exchange or triggering actions.

In order to tackle the contextualised limitations, several scientific challenges need to be overcome, such as the device and service discovery, the cross-platform user authentication, the multi-device context and content synchronisation, and the cross-device application adaptation. The work presented in this paper is part of the research activities that are being performed in the MediaScape European project[1] that addresses the aforementioned challenges. However, this paper is focused on a deep analysis of the challenges for multi-device adaptation and provides a solution for the adaptation domain that enables openness, expressiveness and autonomous behaviour.

Facing the multi-device adaptation challenge means to evolve from developing different applications for specific target devices, controlling one by one their functionality, their interfaces and creating specific event-driven mechanisms for interdevice communication, to a single application where developers describe the complete functionality, interface and the behaviour of the application for a multi-device environment once. The seamless translation of a single application into a multi-device execution while still providing a well fitted portion of the application on each device, is the multi-device adaptation challenge we aim at addressing in the paper.

The main contribution of the work described in this article is the design and implementation of a distributed adaptation architecture for media-driven multi-device applications. We provide a distributed architecture where the application developers create a Web application in terms of logic components once and add properties to define the multi-device behaviour of the application on a high abstraction level. Thus, during the development phase, the application developers do not need to be aware of a particular multi-device context or define views for a specific set of devices (TVs, radios, mobiles, laptops, etc.). Developers have to define global hints to allow the architecture decide the best configuration for visualisation through a dynamic set of involved devices. The libraries and tools comprised in our approach are responsible for analysing and managing the multi-device context of the application and the properties defined in the application to: a) decide which logic components are presented on each device on a specific

[1] http://mediascapeproject.eu

moment, b) create a responsive layout, seamlessly adapted to each device, and c) generate a seamless inter-device communication channel. The user would then be able to smoothly move parts of the application from one device to another in an intuitive manner, and the application would self-adapt to each device.

The methodology followed to achieve the expected contributions is reflected across the different sections of this paper. In Section 2.2.2, the Hybrid TV ecosystem is analysed in order to foresee the experience that next generation multi-device media applications could offer to users. The related work is described in Section 2.2.3, including an analysis of broadcast-related specifications to add companion devices to the TV, specifications for the description of ubiquitous media presentations, and Web mechanisms to enable adaptive application. In Section 2.2.4, the paper focuses on the definition of the adaptation challenges, exploring different scenarios extracted as subsets of the general overview depicted in Section 2.2.2. Section 2.2.5 presents the core contribution of the paper. The section presents a general architectural vision, followed by the definition of specific adaptation design objectives. The section also provides details about the architectural design of the proposed distributed adaptation solution and its advancement beyond the presented related work. Section 2.2.6 presents a validation of the achievement of the design objectives by means of a proof-of-concept implementation of the proposed architecture. Finally, conclusions are stated on Section 2.2.7.

### 2.2.2 The Hybrid TV ecosystem

This section provides pivotal conclusions, extracted from market surveys and reports, to be the base for the definition of a general overview of the Hybrid TV and multi-device experiences.

Hundreds of apps in the various market places aid consumers in controlling the first screen, discovering new content, enhancing their viewing experience and enabling a shared, social experience. These Hybrid TV and multi-device applications are completely oriented to a target device and lack a real interoperability between devices. As mentioned before and addressed in the following section, there are limitations on the standards and technologies in the current state of the art to allow broadcasters and Web application developers to create this kind of application with universal compatibility in a sustainable way.

Due to the success of the Smart TV market there is an increasing demand of multi-device applications. As many HCI researchers increasingly point out, Mark

Weiser's vision of ubiquitous computing [Wei91] will be applied based on multiple devices (tablets, smartphone, TVs, PCs, etc.). The real power of the concept comes from the interaction between devices and sharing experiences with other users.

In order to gain leverage from ubiquitous computing to multi-device applications, it is essential to advance in four areas that span both application developer needs and user expectations. The areas are as follows:

1. security, addressing authentication on multiple devices [EBU14] and privacy of data transferred during the shared experiences;

2. discovery and pairing, associating the available resources: users, devices and services [ZTMO13] [Zie13] [ZMT+14] [DIB+16];

3. synchronisation, providing hybrid broadcast-Internet media synchronisation [BV13] and deploying a shared context available immediately to all participants to provide session awareness and shared timing information for media synchronisation [W3C15a] [ABN13]; and

4. adaptation, spanning autonomous creation of adaptive interfaces suitable to each concurrent viewer, and automatic interface arrangement tracking dynamic application logic.

From all the research areas required to deliver a secure, connected, interoperable and continuous experience over a wide range of devices, this paper focuses on the adaptation domain.

To fully understand the scope of our research and its challenges, the following list summarises the considered contextual factors regarding the trend of media consumption habits of users together with market trends. This is reinforced and aligned with the general overview foreseen by [Cla12] and the accumulated experience from first level broadcasters and technology providers: BBC (British Broadcasting Corporation), BR (Bayerischer Rundfunk), IRT (Institut für Rundfunktechnik), NEC Europe (Nippon Electric Company), Web standardisation bodies like W3C (World Wide Web Consortium) and research institutes like Norut and Vicomtech-IK4 (Visual Interaction & Communication Technologies) [Med14]:

1. Broadcast capable devices are highly relevant for media applications. TV and radio are still the most common devices to consume media services. In

2014 TV dominates the daily media consumption, where most of the content was delivered to the TV set in a traditional manner: over broadcast, cable, satellite or telecommunication connection [Mil14] [Mil14] [Nie14a]. At the same time, radio reaches all people age 12+ during the week [Nie14b]. Radio and TV are in a good position to capture the attention of the audience and drive their interests to specific contents or services.

2. Broadcast capable devices are increasingly connected to Internet. Global Smart TV shipments grew 55% year-over-year where around 50 percent of Smart TV owners across the USA and major European markets are currently using their TV's Internet capabilities [Str14]. Internet radio devices are still immature, evolving to be able to receive and play streamed media from internet radio stations and contents from the home network but often lack a real integration of broadband and broadcast services.

3. Ubiquitous computing centered in the home is a relevant scenario for multi-device media services for end users, broadcasters and application developers. It also brings challenges of ubiquitous computing related to mobility of the users through different contexts and networks, different rooms, multi-member households, going out from home, continuing the experience when arriving at home, etc.

4. Users have other connected devices while watching TV in a home scenario. The consumption habits have changed completely with many TV viewers using tablets, smartphones or laptops while watching TV. According to a study of Ericsson ConsumerLab [An 13] social networking sites and forums are used by 59% of people while watching TV, 49% will use apps or browse the Internet to find out more about the content they are watching, while almost 1 in 3 will discuss what they are currently viewing over social networks or chats.

5. Users are often using two devices simultaneously, accessing to related information in one device to complete the content being watched in the other. Fuelled by the deep penetration rate of smartphones, tablets and laptops worldwide [Gar17], 35% of first screen time is second screened, of which 1/3 is related to the main content [ZTMO13]. Concerning second screen companion experiences, 63% of the consumers accessing synchronised

content on the second screen say it makes them feel a better experience in the shows they are watching [NAP14b].

6. Social experiences relate to a media application. The relation of broadcasted contents and social networks is two way. On the one hand, broadcasted contents are the main driver of social activity at prime time engaging audience to share immediate reactions [Nie14c]. On the other hand, the social media has the ability to engage TV audiences in real-time and in the days that follow. Regarding second screen companion experiences, 14% of second screen time is related to social activity about TV program [Nie14d], chat about a live broadcasted programme, suggesting content, share impressions, etc.

Once the focus of our work has been described, Figure 2.1 presents a diagram of the general overview of the hybrid TV and multi-device experiences. On the one hand, the TV and the radio devices have the broadcast capability to access media content. Moreover, broadcast-related specifications, such as HbbTV, allow the hybrid application signalling through the broadcast channel. On the other hand, the media devices at home, such as tablets, smartphones and laptops, are connected to the Internet. This includes the connected TVs and radios that have the hybrid broadcast-Internet capability creating a bridge between the two channels. Internet brings a way to access applications, content and delivery of application triggers. Accordingly, Figure 2.1 shows all the media devices at home inter-connected between them, offering the user a single experience through multiple displays. On the same way, the figure represents the home-centric *on the move* scenarios, where a user arrives at home with a smartphone and the application adapts to the home context. Finally, the icons on the right side show other homes and users *on the move*, sharing a multi-user experience.

### 2.2.3 **Related Work**

This section provides a summary of the previous work in the field of adaptation for media-driven multi-device applications. It is focused on how the multi-device behaviour of the application is defined as well as how the adaptation is done for a set of devices.

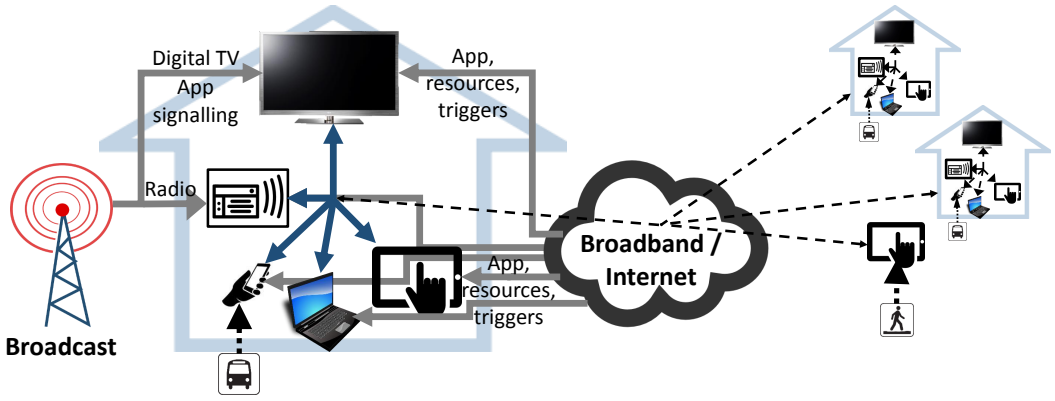


**Figure 2.1:** Diagram of the general overview of the Hybrid TV and multi-device experiences

#### 2.2.3.1 Broadcast-related specifications to add companion devices to the TV

There are broadcast-related standards to provide interactive experiences within Digital TV, providing solutions to add second screen devices to the experience in cooperation with the TV.

Nested Context Language (NCL) [SCMM09] is the declarative language of the Brazilian Terrestrial Digital TV System supported by its middleware called Ginga. NCL is standardised in the ITU-T Recommendation H.761 for IPTV services. As a glue language, NCL relates media objects in time and space without restricting or imposing any media content type, including media objects with imperative and declarative code written using other languages [SMNM10]. Moreover, NCL provides declarative support for presenting distributed DTV applications on multiple devices. Presentation of media objects in NCL applications can be associated to devices using an abstraction called device classes and the parent device, for instance the TV, controls the secondary devices. [SCMM09] presents a solution for the exhibition of multiple devices in cooperation in DTV Systems with NCL.

HbbTV is a pan-European specification published by ETSI to allow broadcasters signalling, delivery through the carousel, rendering and controlling the application with a HTML based browser in the TV or set-top box, providing a hybrid broadcast-Internet experience [Mer11]. In February 2015, the 2.0 specification of HbbTV has been published with new functionalities. In contrast with versions 1.0/1.5, HbbTV 2.0 uses HTML5 and a set of related Web technologies including many modules from CSS level 3, DOM level 3, Web Sockets, etc. Regarding the addition of companion screens to the TV, an HbbTV 2.0 application on a TV can launch an application on a suitably enabled mobile device. More-

over, HbbTV 2.0 enables an application on mobile to remotely launch an HbbTV application on a TV, based on DIAL[2].

These broadcast-related standards, even though they allow mechanisms to launch applications on a companion screen or provide cooperative multi-device application, the application developer needs to create the application completely oriented to a compatible TV. Moreover, developers have to define the behaviour of the application on each device at the development stage, specifying how to visualise the application on the TV and on the second screen. Developers typically need also to handle the communication between the devices, usually limited to a local network communication, for the contextual synchronisation of the experience.

#### 2.2.3.2 Specifications for the description of ubiquitous media applications

Other works deal with the presentation of media applications, describing the different objects and their behaviour, such as SMIL [W3C08], MPEG-4 [PE02] and MPEG-7 [MSS02].

SMIL (Synchonized Multimedia Integration Language) is a W3C Recommendation that specifies an XML format to describe media presentations. It defines a declarative description for timing information to present media, animations, transitions between elements, etc. SMIL 3.0[3] defines the *MultiWindowLayout* module, which contains elements and attributes to describe multiple sets of regions, where multimedia content is presented. However, SMIL does not allow to specify an association between a set of regions and specific devices.

MPEG-4 Part 11 (Scene description and application engine), also known as BIFS (Binary Format for Scenes), defines a spatio-temporal representation of media objects as well as their behaviour in response to interaction. MPEG-4 Part 20 or LASeR (Lightweight Application Scene Representation) is a specification designed for representing and delivering rich-media services to resource-constrained devices such as mobile devices.

MPEG-7 is a multimedia content description standard to provide complementary functionality to the previous MPEG standards, representing information about the content in order to understand its behaviour and see the possibility of interaction with each of the objects and the relation among them.

[2]DIAL Discovery And Launch protocol specification. Version 1.7.2 (2015) http://www.dial-multiscreen.org/dial-protocol-specification

[3]SMIL 3.0. December 2008. http://www.w3.org/TR/SMIL/

However, there is a lack of implementations and solutions on top of SMIL, MPEG-4 and MPEG-7 specifications.

#### 2.2.3.3 Adaptive Web applications and responsive design

There are also available solutions and in-progress language recommendations in the field of adaptive interface of Web applications and its responsive design addressing the adaptation for a single device. However, these approaches do not face the adaptation of an application in the multi-device domain.

Responsive Web design is a key factor in Web development, targeting devices with different features such as connected TVs, laptops, tablets and mobiles. Their heterogeneous displays and characteristics requires the responsive Web design to provide an adequate viewing experience automatically adapted for each device.

HTML features a number of fall-back mechanisms to adjust the content to the device capabilities. Two different approaches to design responsive Web applications can be deployed. On the one hand, a server-side detection based upon information embedded in HTTP requests. The result can be used to return a suitable media content, e.g. a video, where the server selects an optimal bitrate and resolution mode for the requested content based on the device in question.

The second approach for responsive Web applications, covered by the HTML mechanisms, is based on client-side detection and adaptation, enabling a wider set of possibilities. HTML5 includes attributes (*srcset*) that allow application developers to provide multiple resources in varying resolutions for a single image [W3C14a]. Audio and video tags (<*audio*>, <*video*>) permit the definition of different source types and let the browser choose which format to use on the current device (audio/ogg, audio/mp3, etc.). Another powerful client-side mechanism is based on *media queries (@media)* [W3C12]. These simple filters can be applied to CSS styles in order to change the properties based on characteristics of the end-device. Even for showing suitable media content at the client, the trend with streaming protocols like MPEG-DASH moves the adaptation from the server to the client.

Nevertheless, the most widespread way to achieve adaptive interfaces lays on designing the CSS style sheets to be applied to different displays. Therefore, the alternatives often match the application design with the device context, adapting the layout and the user interface. W3C proposes some basic recommendations [W3C10] to improve the user experience of the Web when accessed from mobile devices.

There are several frameworks that support the development of responsive applications. Gridster.js[4] boosts the dynamic grid layouts for drag-and-drop actions based on jQuery. CSS3 Flexbox [W3C14c] empowers the layout definition in the CSS by describing a box model optimised for the user interface design, where the children of a flex container can be laid out in any direction –both horizontal and vertical – in a flexible way. CSS3 Regions [W3C14e] allows elements moving through different regions where CSS Regions library[5] expands the browser support with broader feature coverage. Grid-layout [W3C14d] defines a two-dimensional grid-based layout system, optimised for user interface design. The XML XUL[6] language has been created by Mozilla to design user interfaces. Kontx[7] is the proposal of Yahoo to develop TV apps. EnyoJS[8] is a framework to develop HTML5 apps focused on layouts and panels design. Bootstrap[9] is a framework for developing responsive mobile-first projects on the Web that includes a grid system to scale the layout as the device or viewport size changes.

W3C defined Web Components [W3C14f] to create a Web application in terms of functional components. In comparison with SMIL, it does not have a temporal description of the application but it is a promising emerging standard being widely included in the roadmap of the Web browsers. Web Components include: a) creation of custom elements that can be easily reused in different Web applications including its behaviour; b) import capability of HTML trees to recycle them in other Web applications; c) declaration of DOM (Document Object Model) subtrees as templates ready to be instantiated in the final DOM; and d) management of Shadow DOM, encapsulating and managing DOM trees for the final DOM composition. This high-level block definition from tags to subtrees empowers the recycling possibilities.

There are libraries, such as Polymer[10], that facilitate the development on top of Web Components specification. Furthermore, due to the still limited browser support of this technology, Polymer overcomes the lack of native support of the

[4] Gridster.js website. http://gridster.net/

[5] CSS Regions library, April 2014. https://github.com/FremyCompany/css-regions-polyfill

[6] Mozilla XUL. https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL

[7] Kontx. https://developer.yahoo.com/connectedtv/kontxapiref/

[8] EnyoJS. http://enyojs.com/

[9] Bootstrap. http://getbootstrap.com/

[10] Polymer library's website: https://www.polymer-project.org/

browser with polyfills[11], providing universal browser compatibility. The combination of Web Components and *media queries* boost the responsive design of Web applications and they are core technologies, among others, of the next generation of Google apps [Tec13].

All these languages, recommendations and frameworks for adaptive interfaces and responsive design are very useful for smooth local adaptation but need to be extended towards the multi-device dimension. They are all designed to adapt an application to a device aiming the adaptation depending on the device features. However, they do not consider that an application can be running in one or more devices simultaneously, including only part of the application.

#### 2.2.3.4 **Overview and outlook**

It can be concluded that the described broadcast-related standards provide solutions to create a multi-device experience around a TV. Those standards are limited to a specific combination of devices, such as a TV and a smartphone. Therefore, the application developers need to define the visualisation of the application on the first and second screen at the development stage.

In order to dynamically make decisions on how to split the functionality of a media application across devices, there are specifications with a spatio-temporal description of media presentation, such as SMIL or MPEG-4 Part 11. However, these specifications are not widely deployed.

This differs from the approach of Web Components, which enables reusability and expressiveness on top of HTML5 technologies. Moreover, broadcast-related standards such as HbbTV, are evolving towards HTML5, facilitating the broadcast-Internet convergence. In comparison with SMIL, Web Components lack spatio-temporal data to share timing information on a multi-device context, but Web technologies provide many mechanisms to synchronise state between components [W3C15a] [ABN13].

None of the standards that define interface adaptation and responsive design address applications that span multiple devices. Solutions are designed for the adaptation of an application to a single device, rather than providing the appropriate mechanisms to split the various parts of an application across different devices.

[11] A polyfill is a downloadable code which provides facilities that are not built natively in a Web browser

A more versatile solution is required in order to allow an application to self-adapt automatically to the multi-device context of the user at run-time. For that purpose, the application should be defined in terms of logic components and generic multi-device behaviour. Web Components provide that separation and reusability through components, while taking advantage of widely spread Web mechanisms, thus enabling a complete convergence with the broadcast-related standards, such as HbbTV.

However, the multi-device paradigm creates new challenges to be tackled and requirements to be defined. On the one hand, the components should share a state across the devices for the spatio-temporal and context synchronisation. On the other hand, the Web mechanisms for adaptive Web interfaces, as well as the responsive Web design patterns, need to be extended to the multi-device aspect.

### 2.2.4 Multi-device adaptation challenges

This section presents the multi-device adaptation challenges to be addressed in the proposed architecture. Those are the outcome of the analysis of multi-device scenarios and use cases extracted as subsets of the overview diagram Figure 2.1 described in Section 2.2.2. A more detailed description of the multi-device scenarios and use cases can be found in [Med14].

**Scenario A - Hybrid Live TV Programme**: As Figure 2.2 presents, some of the adaptation challenges can be analysed on a hybrid live TV programme where a broadcaster provides a live sports event. In addition to the 'standard' TV signal, the broadcaster is preparing an application with extra content, such as streaming cameras to follow top athletes or specific spots, maps with geolocated data (e.g. taken from GPS) or real-time statistics.

From a user perspective, users watching sports events on TV can use their mobile devices to find services related to the programme. Some parts can be overlaid on the TV and shared by different households, while other parts can be placed on their mobile devices with more personalised information.

**Scenario B - Multiuser quiz**: Figure 2.3 focuses on a sharing experience between two users, playing a quiz related to a TV programme while on the move. One of them is playing it using the smartphone through 3G, without following the live programme but with all the needed information to play. When arriving home, the TV seamlessly comes into action, providing live media content while the interactive parts remain on the smartphone. Later on, the user moves to the bedroom, only with the smartphone, but connected through Wi-Fi to the Internet.

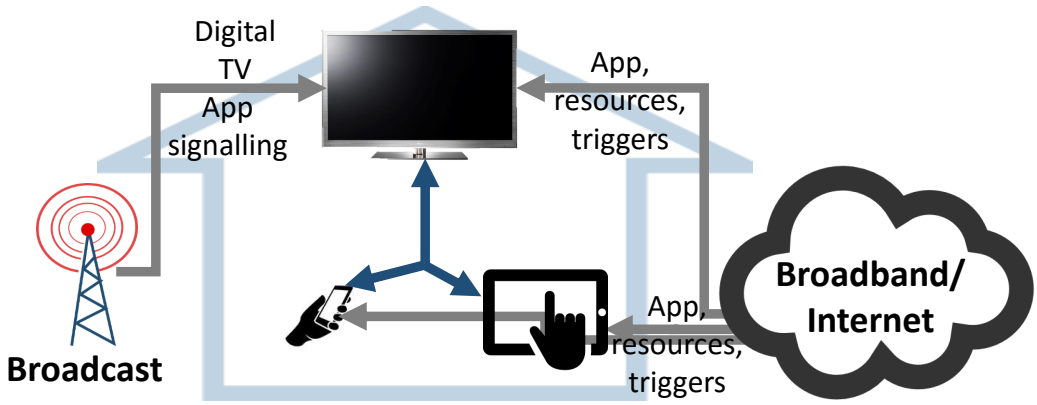


**Figure 2.2:** Diagram for the Hybrid TV Programme

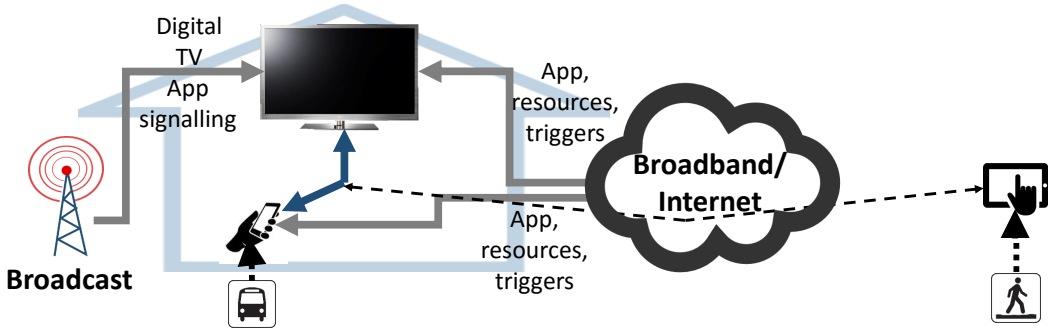


**Figure 2.3:** Diagram for the Multiuser quiz

The application will provide a streaming live video together with the quiz. This scenario covers the companion experiences related to a TV programme, where the rhythm of the application can be synchronised with a live TV programme or with a non-linear experience having an independent timeline.

**Scenario C - Hybrid Radio through a connected device**: The scenario represented in Figure 2.4 shows a user controlling a radio through a laptop and adding bookmarks to contents for later access. In the same way it enables the user to transfer contents between devices, such as moving the audio from one radio to another or moving the controls from the laptop to a smartphone.

Following the process described in [Med14], where scenarios A, B and C are deeply described together with different use cases extracted from those scenarios, we identified a set of challenges to be considered in the design of a multi-device adaptation architecture.[Med14] also provides an analysis of the scenarios across the main stakeholders involved in the scenarios: end users, broadcasters and application developers. In particular, four adaptation challenges have been defined:

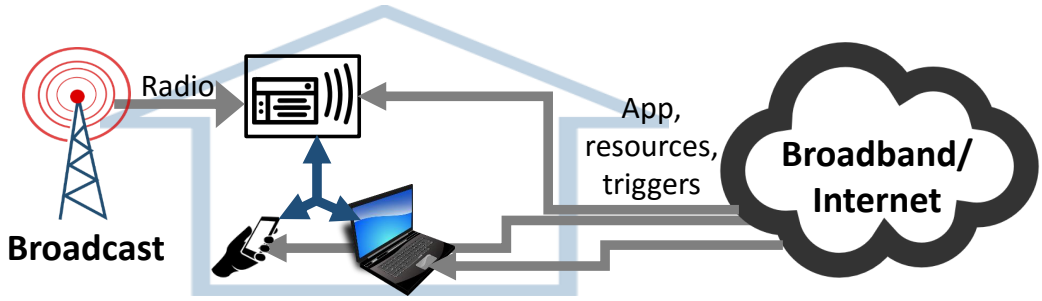


**Figure 2.4:** Diagram for the Hybrid Radio through a connected device

1. **Multi-device adaptation**: The application needs to decide which pieces of content are suitable to be presented on each device, depending the contextual information of the multi-device dimension. Thus, it can provide the user a single experience through one or more devices at the same time. Multi-device context management is required, together with the knowledge of the capabilities of the different devices, so the application has contextual information to take decisions.

2. **User interface adaptation**: Once the multi-device adaptation decisions have been taken, the application has to be shown in a responsive way and adapted to the features of the device (screen size, interaction way, etc.).

3. **Personal and shared device adaptation**: The role of the device in multi-device experiences should be considered to decide which pieces of information are suitable for each device.

4. **Inter-components communication**: The components require a shared context, allowing all components in the application to share necessary state. The shared context must be distributed to all components, regardless of which device they are executing on. This will enable, for instance, a controller component to play or pause a video player, no matter if both components are in the same device or not, or to have a common timeline for different components across multiple devices.

Table 2.1 explores the particular singularities of the defined adaptation challenges in each scenario. **Multi-device adaptation** and **user interface adaptation** are the main challenges that appear in all situations. This is also true for the **inter-components communication** challenge, expressing the need for a flexible state exchange between logic components in order to guarantee the first two

challenges. The **personal and shared device adaptation** challenge has been extracted from the **Hybrid Live TV Programme** scenario, as an important particular case of **multi-device adaptation**.

**Table 2.1:** Analysis of the singularity of the adaptation challenges in each scenario

| A. Hybrid Live TV Programme |
|---|
| Multi-device adaptation |
| • Present the application or its parts depending on the multi-device dimension; TV or TV and mobiles.<br>• Decide the amount of textual information presented on a TV, usually limited.<br>• Avoid parts of the application that may not be suitable for a TV, because the constrained interaction capabilities of the remote control.<br>• Re-structure parts of the application when mobiles are associated providing a richer interaction.<br>• Allow user interaction to move parts of the app from one device to another, such as overlays on TV. |
| User interface adaptation |
| • Present pieces of the application on each device following the responsive design pattern.<br>• Provide a good experience in terms of media consumption and interaction. |
| Personal and shared device adaptation |
| • Decide the adequate information to present depending the nature of the devices.<br>• Information shared by the different users in front of the same TV.<br>• Personal information on personal mobiles. |
| Inter-components communication |
| • Control the application in a single and consistent way through all the devices.<br>• Different household members controlling the same TV. |
| B. Multiuser quiz |
| Multi-device adaptation |
| • Present essential interactive parts when there is only a single mobile device.<br>• Add other components, such as the broadcasted video, when a TV companion joins.<br>• Present streamed video content with high-speed Internet connection. |
| User interface adaptation |
| • Present portions of the application on each device.<br>• Focus interactivity on the mobile and in media content on the TV. |
| Inter-components communication |
| • Communication between components being managed by a single user through different devices.<br>• Communication through components handled by different users, providing a multi-user experience. |
| C. Hybrid Radio through a connected device |

| Multi-device adaptation |
| --- |
| • Identify the best device for a specific activity, like the hybrid radio for a live radio programme.<br>• Use devices with high interactivity capability to control the radio. |
| User interface adaptation |
| • Present the controls in the most appropriate way depending the device: embedded on a Web page on a laptop or full screen on a smartphone.<br>• Provide a good interaction experience to select where to play the audio and how to control it. |
| Inter-components communication |
| • Control the radio from other devices, exchanging state between distributed components. |

The performed analysis shows that the adaptation challenges are common to the different scenarios proposed in this paper. The election of these scenarios has not been arbitrary. On the one hand, the scenarios are *realistic* since the different stake-holders identified them as familiar. On the other hand, they provide a *complete* view of the ecosystem, as combining the different features depicted by Figure 2.2, Figure 2.3 and Figure 2.4, the outcome is the general schema shown in Figure 2.1. Moreover, the scenarios are *non-redundant*, being the three of them necessary to cover the general outline.

### 2.2.5 Web-based distributed adaptation architecture

The Web-based adaptation architecture is the main contribution of this paper, enabling the creation of responsive multi-device media applications that overcome the challenges identified in the adaptation domain. This solution is built over the Web Components standard and deployed on top of a distributed architecture, where each device takes its own adaptation decisions according to the shared context of the multi-device dimension. This section explains the architectural design decisions that have been taken and shows the architecture details focusing on the two main modules to solve the adaptation challenges: the multi-device adaptation engine, addressing the **multi-device adaptation** challenge, and the user interface adaptation engine, facing the **user interface adaptation** challenge.

#### 2.2.5.1 Overview of the architecture

In this Subsection we overview the high-level architecture that has been designed for the development of multi-user and multi-device media centric applications

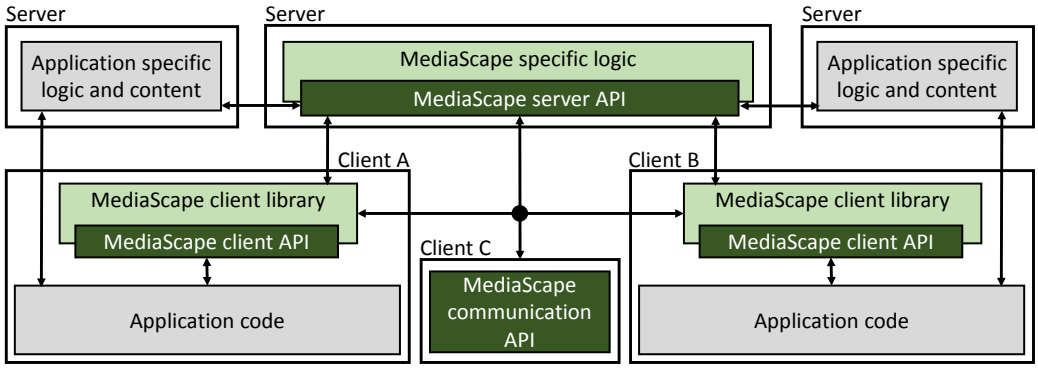


**Figure 2.5:** System overview of the MediaScape architecture

within the MediaScape European project. This architecture aims to solve scientific challenges for such applications, which are wider than the adaptation challenges analysed in this article. Cross-device user authentication, device / user / service discovery and association, and application and content synchronisation are other main research topics together with adaptation in the project.

The starting point of the high level architecture is a client/server approach. Figure 2.5 outlines the general architecture of MediaScape. On the top part of the figure, three server boxes are placed. The application developers, creating multi-device media applications with the MediaScape libraries, do not need technological conditions or requirements for the application specific logic and content on the server-side. This means that the application provider can therefore host the application just as other Web content. The MediaScape toolkit logic is hosted on a second server class (depicted in the top centre of Figure 2.5), providing services to the different MediaScape modules, accessible through an API. The MediaScape services on the server-side are built as modular and open services, making possible to be hosted by the same application provider or by a third party server.

Applications that are distributed to clients are built on top of standard Web technologi es enabling the interoperability with other components of the system. A Web server provides the resources (HTML, JavaScript, and CSS) via HTTP to the application clients. These clients download the required application resources, containing application specific code and MediaScape client libraries. These libraries enable the communication with the required MediaScape services, making them accessible to the application code through a client side API.

Communication between two clients is usually performed through a common application server. Nevertheless direct communication protocols can be used

between applications, such as WebRTC or Bluetooth. Hence, the figure shows a connection between client A and client B.

Additionally, a second client type is included in the architecture (client C). This represents clients, that are incapable of running client side components and therefore cannot benefit from the full MediaScape client library (e.g. the hybrid radio presented in the scenarios). These devices may expose their capabilities with standard interfaces, thus other full MediaScape clients could discover and include it into a multi-device application.

#### 2.2.5.2 **Design objectives**

This section aims to translate the adaptation challenges exposed in Section 2.2.4 into architectural design objectives for adaptation purposes. The required adaptation modules will match the general high-level architecture presented previously.

As an initial design objective, the architecture must provide an **autonomous behaviour** for client devices, enabling a self-organising system in the operational phase. To achieve this, a distributed solution is needed. The alternative centralised design would require a server-side adaptation module orchestrating all the adaptation decisions and delivering what-to-do to client devices. A distributed solution, where the client devices share contextual information and make their own adaptation decisions according to the shared context, follows the Web design trend and removes the need of a dedicated server infrastructure. Distributing the decisions through all client devices removes the possible central server unavailability and assures an autonomous adaptation outcome. The out of date of a client device's shared context, due to any connectivity hazard, will result in a temporary sub-optimal outcome in the worst case.

Moreover, the proposed system has to boost the development of this kind of applications, avoiding broadcasters and application developers to deal with complex and heterogeneous environments, enabling to create multi-device applications by means of a single Web application that simplifies the scientific and technological challenges. In this regard, our approach needs to provide a flexible and modular adaptation architecture, enabling re-use of components and adaptation behaviour from one application to another. A single development environment, with a common set of programming languages based on standard Web technologies, will facilitate the creation of multi-device applications covering a wide range of scenarios. The goal is to avoid ad-hoc developments for a specific adaptation server and for client devices. Thus, the implementation of an

application on top of well-known and existing Web technologies, significantly reduces the learning curve for an application developer. We refer to this feature as **expressive**ness, which is an important design objective.

Finally, the system must follow an open approach to easy the development using, extending or defining standard Web languages, with a potential to be standardised, enabling interoperability with vertically-oriented closed platforms. Accordingly, an **open**ness design objective becomes highly relevant. Clear and open specifications help third parties, application developers and broadcasters to create their adaptation logic or re-use existing generic adaptation rules on their multi-device media applications. Moreover, the research activities presented in this paper, as well as the other parts addressed by the MediaScape project, are being performed in communication and collaboration with different W3C Working Groups. The openness design objective contemplates to follow the recommendations of the on-going discussions and feed them with our outcomes to influence the future recommendations and standards.

The implications of the aforementioned objectives on the design decisions and the system architecture can be summarised as follows:

- **Autonomous behaviour**: The entire application will be delivered to each client device. At run-time each device will know how to behave accordingly to a shared context comprising all the devices used by a user simultaneously. A hierarchical component definition will enable each client device to have a general overview of the entire application in order to load specific resources as required. This strategy follows the lazy-load[12] approach, limiting resource usage on clients, servers and network.

- **Expressiveness**: The media-driven multi-device application will be defined on a common set of programming languages on top of standard Web technologies, and over a single developing environment without custom code on the server side to support adaptation.

- **Openness**: Open specifications over current standard Web technologies providing re-usability and development of new adaptation logics, enhancing those standard Web technologies in a way to likely be included in the roadmap of future standards.

[12] Lazy-loading is a design pattern to defer initialisation of an object until the point at which it is needed, contributing to the efficiency of the application.

As a conclusion, our solution requires a distributed service architecture enabling an autonomous behaviour. It will strongly rely on APIs and service interfaces, and will avoid dependencies from specific, proprietary implementations and/or centralised services, offering expressiveness and openness. The main objective is to open the market of media and service provisioning to all agents, allowing the coexistence of open-source and proprietary solutions through open specifications of module interfaces and data communication protocols.

#### 2.2.5.3 **Architectural design of the Web-based distributed adaptation solution**

The strategy followed for the adaptation mechanism consists in introducing specific components acting as libraries on top of the Web Components standard prioritising the simplicity for application developers and broadcasters. They just have to include an application container (see *app-container* in Figure 2.6) and add all the needed application related components inside. The following code guide describes how to create the application:

```
<link rel="import" href="components/app-container/app-
    container.html">
<body>
  <app-container>
    <component-1></component-1>
    ...
    <component-N></component-N>
  </app-container>
</body>
</html>
```

Figure 2.6 shows the adaptation stack diagram hosted by each client. There are some core blocks for adaptation purposes and others that provide base services. The last ones include a capabilities *introspector* to create self-context (*agent-context*), an *app-context* to provide shared state between all components within a session.

Going deeper in the distributed adaptation core blocks, there are three components. The *component-manager* is aware of the hierarchical definition of the application, taking charge of loading the necessary resources following the lazy-load approach. It orchestrates and manages the fired events between the different components. The *adaptation-engine* and the *UI-engine* are in charge of solving the two main adaptation challenges described in section 4: **multi-device adaptation** and **user interface adaptation** respectively. The nature of

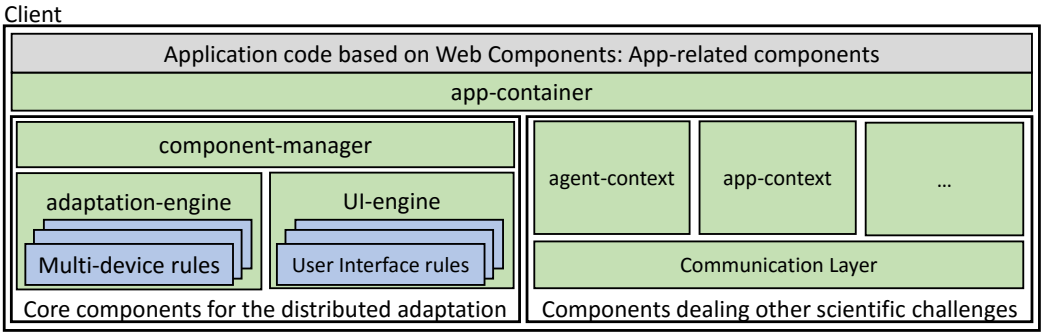


**Figure 2.6:** Client-side block diagram of the Distributed Adaptation Architecture

the devices regarding the personal and shared device adaptation challenge is also overcome by the *adaptation-engine* as a particular case of the multi-device adaptation. Finally, the **inter-components communication** is managed by the *component-manager*. It publishes the services provided by the components to the *app-context*, making other components able to invoke these services even if they execute it on another device.

The adaptation engines apply rules according to the context information and some properties defined in the application code. These properties are aggregated by the application developer as CSS style properties in the following way (both for a specific component and for the main application):

```
<template>
  <style>
    component-1 {
      adaptation-engine-property: value;
      UI-engine-property: value;
    }
  </style>
</template>
```

Figure 2.7 presents the workflow of the two-level adaptation engines. The *adaptation-engine* will automatically apply the multi-device rules on each client device when there is an event on the multi-device dimension of the context. According to these rules, each client device will autonomously determine the set of components to show from the complete list included in the application.

The *UI-engine* will dynamically apply the UI rules to organise the layout for the list of components coming from the *adaptation-engine*. The *UI-engine* will add an abstraction level to organise the interface on top of existing responsive design mecha-nisms such as CSS and *media queries*. Therefore, the *UI-engine*

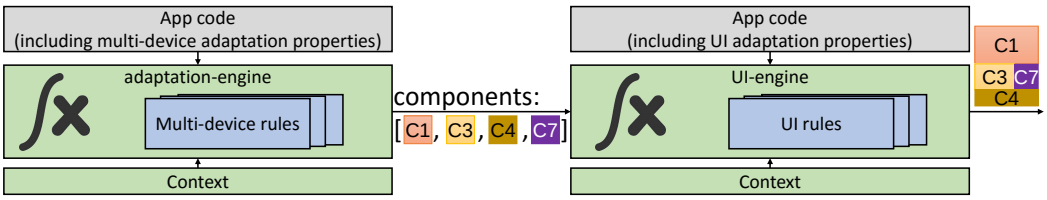


**Figure 2.7:** The workflow and the decisions of the adaptation engines

will re-assess the rules for each incoming event on the multi-device dimension of the context, as well as for each fired browser resizing event on the client device, such as transitions from landscape to portrait.

The following subsections describe the most relevant core adaptation components: *adaptation-engine* and *UI-engine*.

#### 2.2.5.4 **Adaptation engine**

The *adaptation-engine* manages the multi-device behaviour of the application depending on the number of client-devices that a user is using simultaneously and the multi-device rules that the application developer has defined.

For common applications, developers can import existing generic rules on the *adaptation-engine*, and these rules will provide them the default properties that must be specified on the application. Anyway, if developers want to create custom properties, or specific algorithms, the distributed architecture is open to load new rules where custom tags come into the equation.

To overcome the **personal and shared device adaptation** challenge, the developer can define a property on each component establishing if that component is suitable for a shared device or not. This way, the *adaptation-engine* can take correct decision depending on the personal or shared nature from the device context.

More generally, and to overcome the **multi-device adaptation** challenge, the *adaptation-engine* will apply the rules on top of a shared context, where other components - not related directly with the adaptation (see Figure 2.6) - publish the capabilities and features of each device. Therefore, the application developer can define specific mandatory capabilities for a component (e.g. a component requires geolocation information or broadcast capabilities to be available in a device) or suitable capabilities (like arranging one component in the biggest screen and highest resolution, and a second component in a touch screen to allow a better interaction).

As previously presented, these properties will be added as CSS style properties using an intuitive namespace that the *adaptation-engine* rule will use. Section 2.2.6 provides an implementation of a multi-device adaptation rule.

#### 2.2.5.5 **UI engine**

The *UI-engine* dynamically creates a responsive layout organising the output components provided by the *adaptation-engine*. The *UI-engine* applies rules according to the UI properties that the application developer defines for each component and for the application itself. The rules take into account parameters from the context information, such as the features of the device (screen size, capabilities, etc.) and decide, based on templates, the spatial scheme of the components. The layout is generated over CSS properties including *media queries* to integrate responsive design mechanisms.

The *UI-Engine* provides an abstraction layer, adding logic for the organisation of components, benefiting from the context information available. It facilitates the combinatory towards the expressiveness of the UI rules based on templates such as the examples presented in Figure 2.8. When application developers are facing a single-device user interface, they define CSS templates to organise the items in the layout, usually creating a different template for each target device.

However, when application developers are dealing with multi-device applications, this approximation becomes unapproachable. For instance, for an application with 6 different items, within a multi-device scenario, a single device could show all the components, only 5 of them, 4 of them, etc. Furthermore, when for example 4 items are shown on a device, the combinatory of selecting 4 components from a total of 6 rises 15 different options (see equation 2.1 where n is the total number of items of the application and k is the number of items shown in the device, in the example $k = 4$ and $n = 6$). As a result, an application of 6 items has 64 different combinations to create a layout template for each target device [ZTMD15].

$$f(n,k) = \frac{n \cdot (n-1) \cdot (n-2) \ldots (n-k+1)}{k \cdot (k-1) \ldots 2 \cdot 1} \tag{2.1}$$

The current implementation of *media queries* in Web Components allows use of *media queries* both for the main application and for each component. *Media-queries* fire events continuously on detected changes in the window size to re-assess visual properties. In our case, the *UI-engine* meets additional conditions since the *adaptation-engine* can decide to add or remove a component

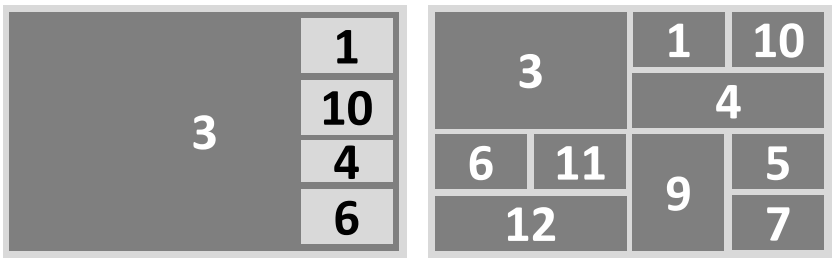


**Figure 2.8:** Different organisations of a layout based on components. On the left a full screen component with picture-in-picture components on top of it is depicted. On the right, the figure presents a fixed layout creating a matrix of components based on their size and aspect ratio.

from that device (e.g. a new device is connected or disconnected in the session). Under these circumstances, the *UI-engine* re-organises the components in the layout, and each component applies the *media-queries* accordingly to the new assigned position and size.

It is important to highlight that discussions have started in W3C to enable custom *media queries* in CSS based on scripts [W3C14b]. The *UI-engine* could take advantage of this feature when it becomes available, as well as fuel the discussion with specific requirements extracted from future steps in our research. Section 2.2.6 presents an implementation of a user interface adaptation rule.

### 2.2.6 **Validation**

This section presents the validation of the defined design objectives over a proof-of-concept implementation of the proposed Web-based distributed adaptation architecture. Our implementation develops all the client-side components described in Figure 2.6 using the Polymer library on top of the Web Components standard. This includes core components for the adaptation (*component-manager*, *adaptation-engine* and *UI-engine*), components dealing with other scientific challenges to provide information to the adaptation engines (*agent-context*, *app-context* and a communication layer to reach the available services) and the *app-container*.

In order to validate the proposed distributed architecture, two sample rules have been developed: one for multi-device adaptation and another one for user interface adaptation. The validation has been done based on the **autonomous behaviour**, **expressiven**ess and **open**ness design objectives.

#### 2.2.6.1 Multi-device adaptation rule

We have implemented a multi-device adaptation rule for validation purposes based on the capabilities of each client device provided by the *agent-context* component (see Figure 2.6). The capabilities are shared across all devices in the session by the *application-context*. However, both context modules are out of the scope of this paper.

The adaptation rule requires the application developer to specify the behaviour of the application using the following custom CSS properties for each component:

- ***adaptation-engine-needs***: a list of mandatory features in the client device for the component, such as the broadcast capability, geolocation information or a camera.
- ***adaptation-engine-best-fit***: a list of desirable features in the client device. For the implemented rule we use *big-screen* considering the resolution and the size of the display and *touchable* regarding the interaction capability of the device.
- ***adaptation-engine-behaviour***: it defines the replication behaviour for each component:
  - ***movable***: the application allows the user to interact with the component and transfer it from one device to another.
  - ***duplicable***: the component can be shown in more than one device at the same time. If it is not duplicable, the component will only be shown in one client.
  - ***required***: the component is indispensable at least in one device, no matter the multi-device context. If it is not required, the component will be shown whenever the required conditions can be satisfied through an established threshold.

The rule is applied autonomously for each client device on run-time. This means each client takes the decisions by itself considering shared context information coming from all clients connected to the session. Figure 2.9 shows the algorithm of the multi-device adaptation rule based on these properties. The movable property has been used to automatically overlay a button in those movable components enabling to move them to the other available devices.

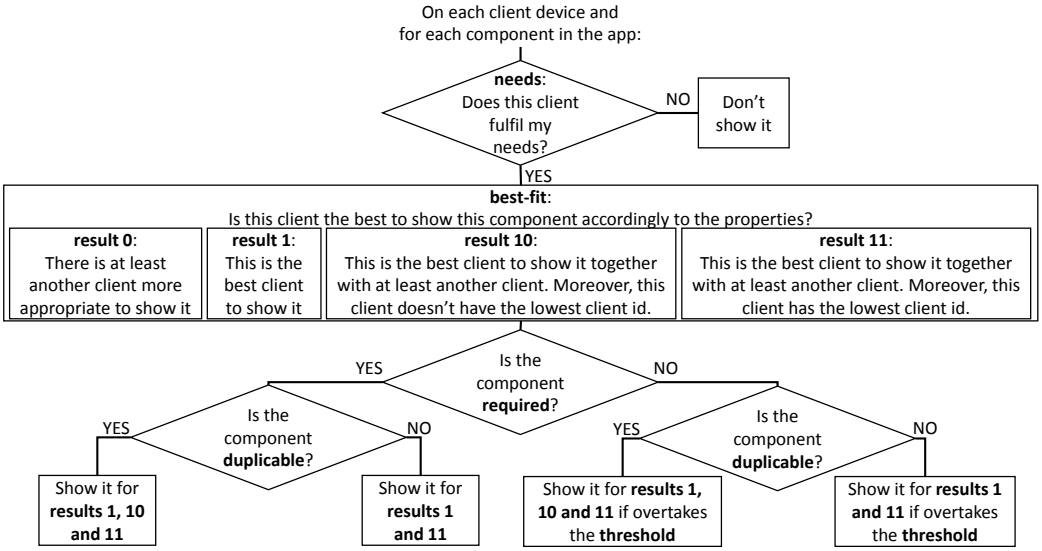


**Figure 2.9:** Multi-device adaptation rule based on the multi-device context of the client device

#### 2.2.6.2 **User interface adaptation rule**

We have also implemented a rule for the *UI-engine* for validation purposes. The proof-of-concept implementation includes the two templates depicted in Figure 2.8: *picture-in-picture* and *fixed* templates. The rule requires the following UI properties for each component:

- ***UI-engine-priority***: the priority of the Web Component inside the application logic. The lower number, the highest priority.
- ***UI-engine-aspect-ratio***: the width and height coefficient that the Web Component needs to maintain its look. It is not mandatory to set an aspect ratio if it is not important for the component visualisation.
- ***UI-engine-size***: the size for a Web Component to show it properly. It is possible to define only the width or the height, setting a specific value in pixels, or defining a *maximum* size property instead a specific value.
- ***UI-engine-template***: the default template to be used. In this case *picture-in-picture* or *fixed*.

On top of these UI properties, the rule determines which template to be used for each client device. For our proof-of-concept implementation, we decided to use the *picture-in-picture* template when the *agent-context* discovers we are on a


**Figure 2.10:** A setup of five different devices being used simultaneously during the performed tests

TV, PC or laptop. However, for smartphones and tablets the *fixed* template will be used. If the *agent-context* is not able to discover the type of device it will use the default template defined in the properties. Depending on this first step, the rule uses the other UI properties (priority, aspect ratio and size) to organise the components inside following the selected template.

For the *picture-in-picture* template, the highest priority component adopts the full screen position while the others are added in order or priority from top to down. We set a specific width and set the height accordingly, obeying the aspect ratio if it is defined or creating a square component.

In the case of the *fixed* template, the rule creates a matrix depending on the window size. Following the priority of the component, the first empty slot with enough space to show it will be assigned, according to the defined size. The engine iterates across the components with a priority-based order, and goes over the layout matrix from left to right and from top to down, filling empty slots with unplaced components. The components without a specific aspect ratio or fixed width and height provide more flexibility to create the layout.

#### 2.2.6.3 **Testing**

The two aforementioned rules have been tested successfully in an application using 3 to 6 different Web Components, inspired by the Hybrid Live TV Programme scenario: three video components, the controller for the videos, a component to visualise metadata information and a map with the location of an object in the video.

The tests have been accomplished using from one to five devices simultaneously. All devices were running the Google Chrome Web browser. Figure 2.10 shows one of the setups used during the experiments. The figure presents a TV showing the main video component with the front view of a train. The laptops on the left and right sides display other video components, with the left view and the rear-right view of the train, respectively. The tablet shows where the train is located in the map, while the smartphone provides two components: information about the next station (name and estimated arrival time), and the play/pause controls. All the content is synchronised, providing interactivity to the user, being able to pause and resume all views or jump to a specific location in the map influencing all the views. Figure 2.10 shows a screenshot of a video[13] that introduces more details about how the application adapts to the multi-device dimension when devices are connected and disconnected during the session.

The testing activities presented in this paper are exclusively focused on the adaptation domain. However, the proof-of-concept implementation integrates other modules in order to address other scientific challenges involved in multi-device application on a hybrid TV environment, such as discovery of the devices and their features in the session, association between devices, and synchronisation [W3C15a] [ABN13]. These modules, together with the adaptation activities, are being developed within the MediaScape project. The quantitative performance analysis of the system is outside of the scope of this paper, since the proof-of-concept implementation only seeks to validate that the multi-device adaptation design objectives are achieved, not the performance of all the integrated modules.

Regarding adaptation, the solution demonstrates automatic tracking of dynamic setups as clients come and go. The proof-of-concept implementation confirms the **autonomous behaviour** of each client device due to the distributed design of the adaptation architecture, since each device runs an instance of the same application. Consequently, the same adaptation rules are executed autonomously for each one of the clients when events are raised (e.g. a new device connected/disconnected on the session). It provides a consistent experience across multiple devices, behaving as expected, according to the properties defined by the application developer and the implemented rules.

From a broadcaster or application developer perspective, the performed tests also conclude that in terms of code and application maintenance, the pro-

[13] Video of one of the performed experiments http://vimeo.com/113514286

posed approach provides simpler development through **expressive**ness. For the proof-of-concept implementation a single application was created. The application-related components have been developed following the Web Components specification. The adaptation JavaScript library and the aforementioned rules were also included, being re-usable from on application to another as they are not necessarily application specific. Our approach extends with CSS-style properties the necessary aspects to specify the multi-device adaptation behaviour to apply the rules on top of Web technologies. Alternatively, should our proposal not be used within the current ecosystem, we would have to pre-define how to separate the views of each one of the screens as part of the development phase. For example, by creating separate HbbTV and mobile applications, while using an event-driven server to manage the communication between applications [ZTMO13] [Zie13]. This increases the specific code for the application, hinders reusability and complicates maintenance.

We also validated the **open**ness of the approach by exploring further the interoperability with the Connected TV. The proof-of-concept implementation presented in the aforementioned video was running on top a Chrome Web browser in a Cozyswan Mk808b Android 4.2 TV Box. An HbbTV 1.5 device was also tested: Panasonic TX-42ASW654 LED TV. Although this device presents some difficulties to perform some of the HTML5 dependant functions, such as some parts of the CSS Grid Layout polyfill, a basic implementation does however run on this device. The implementation of the video components is responsible for creating the HbbTV video object when they are in an HbbTV device. The adaptation proposal presented in the paper decides where to show each component and how to arrange the layout on each device, but does not address the implementation of each component. Moreover, the multi-device adaptation rule also needs to be slightly modified when an HbbTV device is involved to avoid showing more than one video component in the HbbTV device, as usually only one single video object can be managed by HbbTV devices. The architectural design through independent core adaptation components (as managers and adaptation engines) and adaptation rules (both multi-device adaptation and user interface adaptation rules), grants the openness feature, simplifying the process of adding new rules, such as showing only one video component on an HbbTV application.

In order to evaluate the hybrid broadcast-Internet convergence of the solution during the experiments performed with the HbbTV device, the application was launched on a TV using the application signalling of the HbbTV specification.

Other devices, such as smartphones and tablets, were associated afterwards following the experience of the Hybrid Live TV Programme scenario. To integrate the broadcasted video as one of the video components, we extended the multi-device adaptation rule to set broadcasting capabilities to a component in the *adaptation-engine-needs* property. Therefore, the broadcasted video will only be shown on a TV unless a streaming alternative is provided. The broadcast-Internet media synchronisation is out of the scope of this paper but addressed in [BV13] and [W3C15b].

As a summary, these experiments demonstrate our adaptation architecture proposal, giving a proof-of-concept implementation that fulfils all requirements, adaptation challenges and design objectives. The viability of the proposed distributed adaptation architecture has been validated for the Hybrid Live TV Programme scenario. It presents open mechanisms to aggregate new rules with new multi-device adaptation algorithms to make decisions, run new user interface templates, and create custom logic to decide when to use the appropriate template on each device. All these features validate the feasibility of the scenarios presented in Section 2.2.4 through the proposed architecture.

### 2.2.7 Conclusions

This paper proposes a Web-based distributed adaptation architecture for multi-device applications driven by media content. This interoperable and standard approach allows broadcasters and media application developers to easily create this kind of applications by extending the Web Components standard. It also provides broadcast-Internet convergence through Web technologies, following the trend of the broadcast-related standards for the delivery of hybrid services, such as HbbTV. Moreover, the proposed approach improves the process that broadcasters and application developers currently need to implement, distribute and maintain over a set of complex technical solutions tailored to each specific target platform.

From the diverse scientific challenges involved in multi-device application creation, our architecture is focused on the adaptation domain. We lay the foundation of some principles of the hybrid TV ecosystem based on media consumption habits and market trends, and on the existing related work. On top of that, we identify the adaptation challenges extracted from the analysis of different scenarios and use cases.

In order to meet the adaptation challenges we propose a distributed adaptation architecture that provides a unified methodology and a common specification over Web Components, extending the current state of the art towards the multi-device dimension within the Hybrid TV environment. The presented architecture enables the development of multi-device applications offering users coherent experiences across multiple devices. The solution enables the following aspects:

- A distributed architecture that allows each client to make autonomous decisions according to a shared context and still provide a consistent view to the user through multiple displays simultaneously.

- A single development environment for broadcasters and application developers that facilitates the creation of multi-device applications without dealing directly with all the adaptation challenges and possible context situations, enabled by the expressiveness of the approach on top of Web technologies.

- The design of generic or custom adaptation rules to create multi-device responsive applications, boosting the re-usability and the development of this kind of applications, due to the openness of the solution built on top of standard technologies.

This paper presents a proof-of-concept implementation of the architecture, including some adaptation testing rules to validate the design objectives of the architecture and its modularity and flexibility. These experiments demonstrate distributed decision making with autonomously executed adaptation engines, tracking dynamic setups automatically and performing real-time self-organising. The experiments also validate that the proposed scenarios can be achieved by creating adaptation rules on top of our architecture. The results reflect the openness of the proposed approach on top of Web Components and the expressiveness of the properties added for adaptation purposes built over standard technology.

In essence, we propose a distributed architecture for the creation of adaptive multi-device media applications using standard and interoperable Web technologies and extending those in a way to be standardised.

#### 2.2.7.1 Acknowledgment

The MediaScape project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement 610404. The contribution from the UPV/EHU has been supported by the Basque Council under grant agreement IT395-10. The first author would like to thank Iñigo Tamayo, Angel Martin, Ana Dominguez and Igor G. Olaizola, researchers in Vicomtech-IK4, for supporting the architectural design and implementation presented in the paper.

## 2.3 User interface adaptation for multi-device Web-based media applications

- **Title:** User interface adaptation for multi-device Web-based media applications
- **Authors:** Mikel Zorrilla, Iñigo Tamayo, Angel Martin, Ana Dominguez
- **Conference:** Broadband Multimedia Systems and Broadcasting (BMSB)
- **Publisher:** IEEE
- **Year:** 2015
- **DOI:** http://dx.doi.org/10.1109/BMSB.2015.7177251

**Abstract** - The quest to transform the television viewing experience into a digital media service is happening thanks to the addition of companion screens to the TV. Multi-device experiences become more intuitive and easier to use federating cooperative devices. They also bring new creative opportunities to schedule and distribute interactive content synchronised with the TV programme through any connected screen. The rise of HTML5 to develop responsive applications across multiple devices adds a significant amount of improvement enabling universal delivery. A key challenge to harness the power of navigation engaged with the story on the TV is the responsive design of a unique application spanning all the available screens. This paper presents user tests in order to explore the relevant parameters to create responsive User Interfaces for Web-based multi-device applications driven by media content.

**Index Terms** - Multimedia systems, Digital multimedia broadcasting, Interactivity, Future services of Broadcasting, Pervasive computing, Ubiquitous computing, Context-aware services, new human-device interaction.

### 2.3.1 Introduction

Companion experiences engage consumers in relevant contents on second devices -smartphone, tablet or laptop- while watching something in the first screen -usually a TV-. The industry is adopting second screen viewing services to capture audiences and deploy new monetisation models.

Web-based applications enable universal delivery via HTML5 to reach those users who do not want to download an application. This trend is also driving the evolution of HTML5, making a wider range of devices capable of running applications that gain features previously available only through native SDKs [Ant12]. Connected TVs are also following this trend towards Web technologies. HbbTV 2.0 [Mer11] uses HTML5 and a set of related Web technologies including many modules from CSS level 3, DOM level 3, Web Sockets, etc. and provides mechanisms for the addition of companion screens to the TV.

Once the self-capacity and interoperability are being addressed by HTML5, the key challenge for next generation applications is to provide users coherent multi-device experiences with simple and intuitive interfaces that ease the navigation through the information provided with a right timing. This is a natural step in the adaption of the market and society to the growing behaviour of users accessing services from several devices simultaneously [CD12], aiming to have a single experience through multiple devices at the same time.

Despite the clear opportunities to catalyse new business models, the lack of standards hinders the creation of seamlessly connected, intuitively converged and conveniently continuous experiences across a heterogeneous ecosystem of devices. The priority for future research is shifting towards fully manageable, context-sensitive and controllable or self-regulating multi-device applications.

To overcome all the intrinsic features needed on a distributed web application, a complete capabilities stack is needed. It should comprise: a discovery service layer to federate other experience participants; a cross-platform user authentication layer for security; a communication layer to consolidate a synchronised multi-device context by means of autonomous information exchange and event triggering; and a cross-device application adaptation to distribute the experience across all the devices and suit the different visual components to each specific device conditions. The work presented in this paper is part of the research activities that are being performed in the MediaScape European project[14] , which

[14] http://mediascapeproject.eu

addresses the aforementioned challenges. In this case, this paper focuses on the User Interface adaptation challenge for media applications across multiple devices.

To really boost the interface creation and maintenance of multi-device Web applications, it is mandatory a universal mechanism that, based on Web standards, considers common possible situations that fire a set of orchestrated adaptation actions. Thus, it eases design and reusability while provides a default behaviour valid for a wide range of applications and contexts.

To this end, it is necessary to evolve application design from developing different applications for concrete target devices and application roles, adding to each one ad-hoc mechanisms to control one by one their functionality.

The multi-device adaptation challenge addresses the seamless translation of a single application into a multi-device execution, providing a well fitted portion of the application on each device. And providing a responsive User Interface on each device is the goal of the research presented on this paper.

This approach differs from the current mechanisms to provide second screen experiences with standards such as HbbTV. [ZTMO13] and [Zie13] provide a solution where different applications need to be developed for the HbbTV device and for the mobile, while an event-driven server deals with the communication between them.

This paper presents user tests performed over 47 end users to analyse the impact of different parameters on how to arrange the User Interface for simultaneous Web-based multi-device media applications. These conclusions will help multi-device application developers to design a unique multi-device media application that will provide a coherent experience across multiple devices at the same time, with a responsive User Interface on each device depending on the parts of the application that is presenting regarding the multi-device dimension context.ç

### 2.3.2 State of the art

This section provides a state-of-the-art of the existing technologies and frameworks to create responsive and device-adapted Web applications. Although there are available solutions and in-progress language recommendations in the field of adaptive interface of Web applications, these approaches are all designed for local adaptation, without considering that the application can be running in more than one device simultaneously.

W3C defined Web Components [W3C14f] with the aim of modularising Web applications. Web Components are made up of four valuable elements which empower recycling possibilities:

- ***Custom elements***: personalized elements easily reusable in different Web applications including its behaviour.
- ***HTML imports***: capability of HTML trees to include and reuse them in other Web applications.
- ***Templates***: declaration of inert DOM (Document Object Model) subtrees ready to be instantiated in the final DOM.
- ***Shadow DOM***: for maintaining functional boundaries between DOM trees and its interactions with each other within a document and enabling a better functional encapsulation.

However, native support of the browser is still limited. Therefore, there are some libraries, such as Polymer[15] , that facilitate the development on top of Web Components specification and overcome the support lack with polyfills[16], providing universal browser compatibility.

Web Components provide a logic component definition of the Web application to make multi-device adaptation decisions on top of those. However, due to the enormous variety of existing devices, such as connected TVs, laptops, tablets and mobiles, Responsive Web Design (RWD) is one of the techniques that takes the lead in Web development. RWD returns a Web application able to adapt and change itself dynamically depending the target device, e.g. the capabilities of the device such as display size or resolution, providing a rich viewing experience.

To that end, HTML features a number of fall-back mechanisms. In order to obtain a responsive Web application, two different approaches could be followed. On the one hand, if a suitable media content is needed, e.g. a video changing his resolution and optimal bitrate mode for each device, a server-side detection could be used, where the server is responsible of selecting the right content, basing its decision on the embedded information in HTTP requests.

On the other hand, there is a client-side approach covered by the HTML mechanisms, where detection and adaptation happen at the client, allowing a wider

[15] Polymer library's website: https://www.polymer-project.org/

[16] A polyfill is a downloadable code which provides facilities that are not built natively in a Web browser

range of possibilities. Here, HTML5 attributes (*srcset*) play an important role, giving the possibility of providing multiple resources varying resolutions for a single image [W3C14a]. A similar fact is also possible at audio and video tags, where the browser chooses which format to use on the current device after having defined different source types (*video/mp4*, *video/ogg*, etc.). Another powerful mechanism is based on some directives called Media Queries (*@media*) [W3C12]. These directives can be included into CSS style sheets in order to change the properties based on characteristics of the end-device. In addition to the aforementioned, the trend with streaming protocols like MPEG-DASH moves the adaptation from the server to the client.

Thus, Web Components and Media Queries represent a perfect match for the RWD and the core technologies, among others, of the nearby future of Google apps [Tec13]. In this field, W3C proposes some basic recommendations [W3C10] to improve the development of rich and dynamic mobile Web applications.

In this context, there are several frameworks that support and facilitate the development of responsive applications. CSS3 Flexbox [W3C14c] suggests a CSS box model optimised for the user interface design, where the children of a flex container can be laid out in any direction –both horizontal and vertical – in a flexible way. CSS3 Regions [W3C14e] allows elements moving through different regions where CSS Regions library[17] expands the browser support with broader feature coverage. Grid-layout [W3C14d] defines a two-dimensional layout system, where the children of a grid container can be positioned into arbitrary slots in a flexible or fixed grid. Gridster.js[18] is a jQuery plugin to build dynamic drag-and-drop muti-column grid layouts. XUL[19] is Mozilla's XML language to design user interfaces. Kontx[20] is the proposal of Yahoo to develop TV apps. EnyoJS[21] is a framework to develop HTML5 apps focused on layouts and panels design. Bootstrap[22] is a framework for developing responsive mobile-first projects on the Web that includes a grid system to scale the layout as the device or viewport size changes.

All these languages, recommendations and frameworks for adaptive interfaces and responsive design are very useful, but they do not face the adaptation of

[17] CSS Regions library, April 2014. https://github.com/FremyCompany/css-regions-polyfill

[18] Gridster.js website. http://gridster.net/

[19] Mozilla XUL. https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL

[20] Kontx. https://developer.yahoo.com/connectedtv/kontxapiref/

[21] EnyoJS. http://enyojs.com/

[22] Bootstrap. http://getbootstrap.com/

an application in the multi-device domain, including only part of the application in each device.

### 2.3.3 Multi-device model

Our research aims to explore which parameters have a relevant impact on the arrangement of the components to be shown on each device in a simultaneous multi-device application.

We will have a media Web application implemented with different logic parts developed with Web Components, and on top of this, we assume there will be an adaptation engine, out of the scope of this paper, taking the decisions of which components present on each device. For instance, if a user is consuming a media application through different devices at the same time (e.g. a TV and a smartphone), the adaptation engine will decide which components to show on the TV and which ones on the smartphone. But once the adaptation engine decides the components to be shown on each device, a responsive User Interface should be created, able to adapt to the context changes (new devices connected or disconnected by the user).

When application developers are facing a single-device user interface, they define CSS templates to organise the items in the layout, usually creating a different template for each target device. However, when developers are dealing with multi-device applications, this approach becomes unachievable. For instance, for an application with 6 different items, within a multi-device scenario, a single device could show all the components, only 5 of them, 4 of them, etc. Furthermore, when for example 4 items are shown on a device, the combinatory of selecting 4 components from a total of 6 rises 15 different options (see equation 2.2 where n is the total number of items of the application and k is the number of items shown in the device, in the example $k = 4$ and $n = 6$). As a result, an application of 6 items has 64 different combinations to create a layout template for each target device.

$$f(n,k) = \frac{n \cdot (n-1) \cdot (n-2) \ldots (n-k+1)}{k \cdot (k-1) \ldots 2 \cdot 1} \tag{2.2}$$

A more versatile solution is desirable in order to compose automatically a responsive user interface following some patterns provided by the application developer and based in layout templates. This paper aims to explore which parameters are relevant to create a responsive User Interface under these circumstances.

Our hypothesis underlines these four parameters to affect to the User Interface:

- **The device**: As happens in a single-device application, the target device is very relevant to build a responsive Web application. In the same way, the devices involved in a multi-device application are expected also to be relevant.

- **The number of Web Components**: The quantity of pieces of information to be shown in that device can affect on how to present the content.

- **The nature of the application**: This parameter could be important to decide the arrangement of the User Interface. For instance, if there is a main video and related information on the device, or if the video is being displayed on another device and that device is being used only for extra information.

- **Other devices being used at the same time**: We want to know if having a second device being used simultaneously has an impact on how the user wants to arrange the components in the first screen.

### 2.3.4 User tests

To find out the influence of each parameter, we enclosed them to these options:

- **The device**: 3 different devices. A Motorola Moto G Smartphone in portrait mode, A Nexus 10 tablet in landscape mode, and a Samsung UE40C8000 TV.

- **The number of Web Components**: Showing 3 or 6 components at the same time.

- **The nature of the application**: Two different scenarios. At least one of the components is a video, or there is not a video among the components.

- **Other devices being used at the same time**: Two possible situations. The evaluated device is the only one being used by the user or there is another device as a companion screen.

**Table 2.2:** Combinations with the defined parameters

| Id | The device | The number of web components | The nature of the application | Other devices being used at the same time |
|---|---|---|---|---|
| 1 | TV | 3 | At least one video | No |
| 2 | TV | 6 | At least one video | No |
| 3 | TV | 3 | No videos | No |
| 4 | TV | 6 | No videos | No |
| 5 | Tablet | 3 | At least one video | No |
| 6 | Tablet | 6 | At least one video | No |
| 7 | Tablet | 3 | No videos | No |
| 8 | Tablet | 6 | No videos | No |
| 9 | Smartphone | 3 | At least one video | No |
| 10 | Smartphone | 6 | At least one video | No |
| 11 | Smartphone | 3 | No videos | No |
| 12 | Smartphone | 6 | No videos | No |
| 13 | Tablet | 3 | At least one video | Yes, A TV |
| 14 | Tablet | 6 | At least one video | Yes, A TV |
| 15 | Tablet | 3 | No videos | Yes, A TV |
| 16 | Tablet | 6 | No videos | Yes, A TV |

Testing images have been created simulating a broadcasted live F1 race scenario with all the possible combinations of the first three parameters (see Table 2.2 from Id 1 to 12).

Apart from these 12 context situations, we created 4 more to evaluate the "Other devices being used at the same time" parameter. We defined as a second screen a TV showing two fixed components and presented different contexts in a tablet, making the user think about how to present the content in the tablet, while they were also watching related content in the TV. As an outcome we have 4 new combinations in Table 2.2 from Id 13 to 16.

For each one of the 16 combinatorial contexts, we created always four different user interface arrangement patterns:

- **Grid Layout**: Based on the CSS Grid Layout Module Level 1[23] (see example in Figure 2.11a).

[23] http://dev.w3.org/csswg/css-grid/

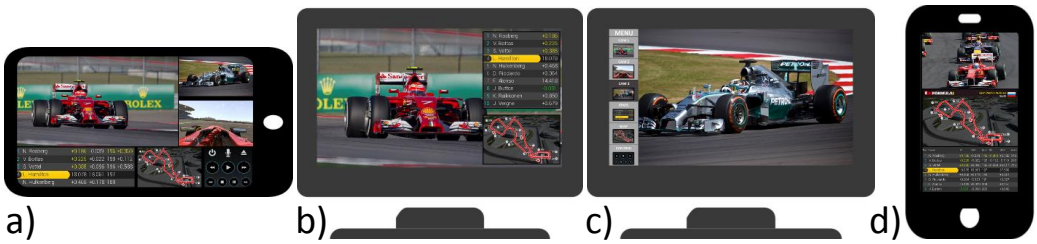


**Figure 2.11:** a) A grid template layout example on a tablet in the context with ID number 6. b) A PiP template layout example on a TV in the context with ID number 1. c) A menu template layout example on a TV in the context with ID number 2. d) A horizontal template layout example on a smartphone in the context with ID number 9.

- **Picture-in-picture Layout (PiP)** (see example in Figure 2.11b).
- **Menu Layout** (see example in Figure 2.11c).
- **Horizontal Layout** (see example in Figure 2.11d).

The tests were done with 47 users, one by one, being always an expert presenting each one of the 16 context situations. The expert gave them a very brief description of the context of the testing and ask them to choose always the layout they would prefer on that moment to see the F1 race. All the tests were carried out in the Digital Home Lab of Vicomtech-IK4, where there is a similar environment on what we can find on a living room. From the 47 users, 40 of them were researchers in Vicomtech-IK4, with expertise on different fields, and 7 of them were administrative staff people. It took around 15 minutes to perform the test with each user, so around 12 hours in total, divided in three different days. Figure 2.12 presents pictures took during the tests.

### 2.3.5 Results of the user tests

Table 2.3 presents the answer that the users gave for each one of the 16 contexts. Moreover, the following sub-sections present the behaviour of each one of the users when one of the parameters changed, in order to evaluate which parameter changed their mind. It measures if the user selected a different layout when the context changed.

**Figure 2.12:** Images from user tests. In the left a user in front of the context situation with ID number 16 with the menu layout on the tablet. In the middle a user with the context situation with ID number 16 with the PiP layout in the tablet and in the right a user in the context situation with ID number 9 with the horizontal layout on the smartphone.

**Table 2.3:** Results of the chosen layouts on each situation

| Id | Results LAYOUT: NUMBER OF ANSWERS \| PERCENTAGE | | | |
|---|---|---|---|---|
| | **Grid** | **Pip** | **Menu** | **Horizontal** |
| 1 | 19 \| 40% | 13 \| 28% | 14 \| 30% | 01 \| 02% |
| 2 | 32 \| 68% | 08 \| 17% | 06 \| 13% | 01 \| 02% |
| 3 | 28 \| 60% | 16 \| 16% | 03 \| 03% | 00 \| 00% |
| 4 | 33 \| 70% | 07 \| 15% | 05 \| 11% | 02 \| 04% |
| 5 | 09 \| 19% | 26 \| 55% | 11 \| 23% | 00 \| 00% |
| 6 | 34 \| 72% | 08 \| 17% | 05 \| 11% | 00 \| 00% |
| 7 | 23 \| 49% | 14 \| 30% | 07 \| 15% | 03 \| 06% |
| 8 | 31 \| 66% | 09 \| 19% | 07 \| 15% | 00 \| 00% |
| 9 | 05 \| 11% | 04 \| 09% | 20 \| 43% | 18 \| 38% |
| 10 | 22 \| 47% | 03 \| 06% | 19 \| 40% | 03 \| 06% |
| 11 | 11 \| 23% | 08 \| 17% | 09 \| 19% | 19 \| 40% |
| 12 | 16 \| 34% | 08 \| 17% | 15 \| 32% | 08 \| 17% |
| 13 | 17 \| 36% | 17 \| 36% | 13 \| 28% | 00 \| 00% |
| 14 | 28 \| 60% | 07 \| 15% | 10 \| 21% | 02 \| 04% |
| 15 | 30 \| 64% | 06 \| 13% | 10 \| 21% | 01 \| 02% |
| 16 | 33 \| 70% | 03 \| 06% | 11 \| 23% | 00 \| 00% |

#### 2.3.5.1 Changing the device

Number of users that changed the preferred layout when the context was changed from smartphone to tablet:

- From id 5 to 9: 38 | 80.85%
- From id 7 to 11: 34 | 72.34%
- From id 6 to 10: 26 | 55.32%
- From id 8 to 12: 26 | 55.32%
- **AVERAGE: 65.96%**

Number of users that changed the preferred layout when the context was changed from tablet to TV:

- From id 1 to 5: 24 | 51.06%
- From id 3 to 7: 14 | 27.78%
- From id 2 to 6: 21 | 44.68%
- From id 4 to 8: 20 | 42.55%
- **AVERAGE: 41.52%**

Number of users that changed the preferred layout when the context was changed from smartphone to TV:

- From id 1 to 9: 36 | 76.60%
- From id 3 to 11: 36 | 76.60%
- From id 2 to 10: 29 | 61.70%
- From id 4 to 12: 30 | 63.83%
- AVERAGE: 68.68%

The overall average impact of changing the device is **59.05%**.

#### 2.3.5.2 **Changing the number of components**

Number of users that changed the preferred layout when the context was changed from an application with 3 components to 6 components when at least one of the components was a video:

- From id 9 to 10: 31 | 65.96%
- From id 5 to 6: 33 | 70.21%
- From id 1 to 2: 23 | 48.93%
- From id 13 to 14: 25 | 53.19%
- **AVERAGE: 59.57%**

Number of users that changed the preferred layout when the context was changed from an application with 3 components to 6 components when there was not a video component:

- From id 11 to 12: 34 | 72.34%
- From id 7 to 8: 23 | 48.95%
- From id 3 to 4: 19 | 40.42%
- From id 15 to 16: 14 | 29.78%
- **AVERAGE: 47.87%**

The overall average impact of changing the number of components is **53.72%**.

#### 2.3.5.3 **Changing the nature of the application**

Number of users that changed the preferred layout when the context was changed from an application with 3 components and at least one of them a video component, to an application with 3 components but without a video component:

- From id 9 to 11: 30 | 63.83%
- From id 5 to 7: 30 | 63.83%
- From id 1 to 3: 27 | 57.44%
- From id 13 to 15: 17 | 36.17%

- **AVERAGE: 55.32%**

Number of users that changed the preferred layout when the context was changed from an application with 6 components and at least one of them a video component, to an application with 6 components but without a video component:

- From id 10 to 12: 20 | 42.55%
- From id 6 to 8: 18 | 38.29%
- From id 2 to 4: 14 | 29.78%
- From id 14 to 16: 16 | 34.04%
- **AVERAGE: 36.17%**

The overall average impact of changing the nature of the application is **45.74%**.

#### 2.3.5.4 **Changing the number of devices being used at the same time**

Number of users that changed the preferred layout when the context was changed from an application in the tablet to the same application in the tablet but having a TV with additional information:

- From id 5 to 13: 25 | 53.19%
- From id 7 to 15: 24 | 51.06%
- From id 6 to 14: 20 | 42.55%
- From id 8 to 16: 18 | 38.30%
- **AVERAGE: 46.28%**

The overall average impact of changing the number of devices being used at the same time is **46.28%**.

### 2.3.6 **Analysis of the results**

After the user tests performed we deduced the following from the results:

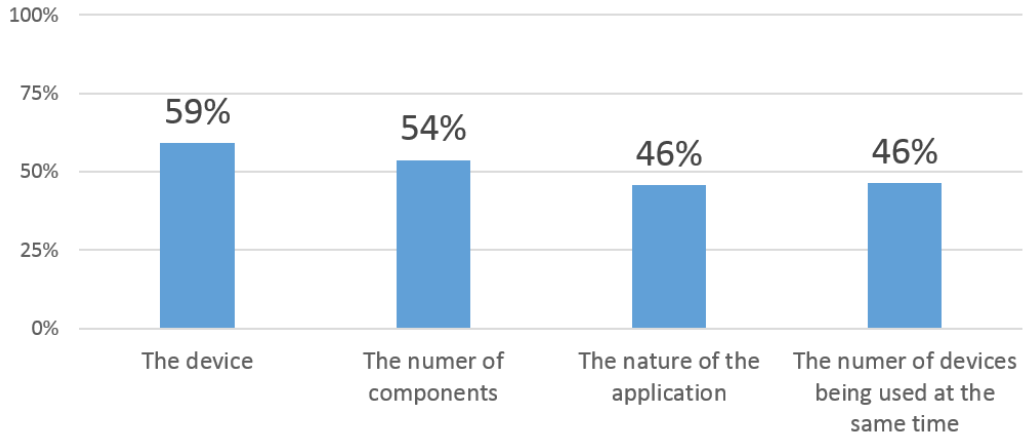


**Figure 2.13:** Impact of the four defined parameters

#### 2.3.6.1 The four parameters analysed as an hypothesis have a big impact on the user interface

Analysing the impact of each one of the four parameters defined to be measured in this prototype, we conclude that the four ones are relevant and have a big impact above 45%. This means that at least 21 of the 47 users tested change the mind when one of the parameters changes in the context.

Figure 2.13 shows that the most relevant parameter is the device, with an impact above 59%. As happens for "single device" application, it was also expected to be an important parameter for multi-device applications, where the user is consuming an application from more than one device at the same time. The number of components parameter is also very relevant being around 5 percentage points below. This means that it is important to change the arrangement of the user interface depending the outcome of the adaptation engine while the number of components to be shown changes.

The nature of the application and the number of devices being used at the same time are also relevant parameters, both of them around 46%, to take them into account to develop the final software of the UI Engine.

#### 2.3.6.2 There is a higher impact on the device parameter if we change the orientation of the device from landscape to portrait

Being 59.05% the impact of the device parameter, it increases considerably when we compare how affects the user a change in the context when moving from a device in portrait (the smartphone in our tests) to a device in landscape (the tablet

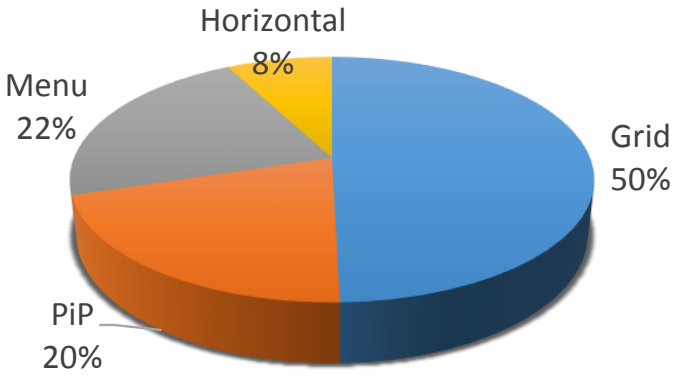


**Figure 2.14:** Percentage of the selected layouts as an average of all the different contexts

and the TV in our tests). In this situation the impact is 67.82%.

On the other hand, when there is a change in the device, but both are in the same orientation (changing from tablet to TV in our tests), the impact of the parameter is around 41%. Still relevant, but more than 25 percentage points below.

#### 2.3.6.3 **The grid layout is the preferred by users**

The grid layout is the most selected by the users from the four possible layouts, with a percentage of 50%. Figure 2.14 shows the average of the selected layouts by the users taking into account all the possible context situations. It is the preferred layout in 13 of the 16 contexts and it is never the worst layout.

#### 2.3.6.4 **The horizontal layout is the most rejected one by users**

As Figure 2.14 shows, the horizontal layout have been chosen in only 8% of the situations, 42 percentage points below the grid layout, and 12 percentage points below the second worst layout (the PiP layout). The horizontal layout is the least selected layout in 14 of the 16 different context situations. However, in those two circumstances where it is not the worst layout (context ID 9 and 11), it obtains a very good result, being the preferred one in context number 11. This analysis gives us the outcome for the next three conclusions.

#### 2.3.6.5 The horizontal layout only has sense with a low number of components and in a portrait device

The results of the horizontal layout show that there could be very specific circumstances where a very specific layout has sense, but won't be useful out of that conditions. So our solutions needs to be open and easy to add a specific user interface for an application by the application developer.

#### 2.3.6.6 Horizontal as a specific case of the grid layout

If we consider the horizontal layout as a specific case of the grid layout, where there is only one column, the grid layout is reinforced as the preferred layout increasing from the 50% to the 58% of the context situations. When the horizontal layout is preferred by the users, it can be considered as a transfer from grid to horizontal, since the sum of grid and horizontal remains stable for all the devices.

#### 2.3.6.7 PiP and menu layouts are more neutral

Rarely are the most preferred layouts (PiP is the preferred one in 2 of the 16 situations and menu is the most selected one in 1 of the 16 circumstances), but almost never the most rejected ones (PiP layout is the most rejected one in 4 of 16 contexts while the menu layout is never the most rejected layout). For a considerable big screen (a tablet and a TV in our tests) the PiP layout is preferred to the menu layout, and for small screens (the smartphone in our tests) the menu layout is preferred to the the PiP.

In fact, for the smartphone the PiP was selected by the 11% of the users while the menu was preferred by the 34% of the users. However, for tablets and TVs, around 26% of the users prefer the PiP while around 16% of the users select the menu layout. The results also make clear that in order to choose only one neutral layout from PiP and menu, the menu layout is more neutral since it is never the worst option for the user and Figure 2.14 shows that it is more selected than the PiP.

### 2.3.7 Conclusions

Existing standards, recommendations and frameworks address the creation of adaptive user interfaces and responsive design of an application. However, they

do not consider the adaptation in the multi-device domain. The trend of consuming media content across multiple devices at the same time is transforming the television viewing experience, where users add companion screens to the TV.

This fact motivated the presented user tests to explore which are the relevant parameters to create responsive User Interfaces for Web-based multi-device applications, where it is not possible to define a specific layout for all the context circumstances and device combinations.

We proposed four parameters to be relevant as a hypothesis in the creating of multi-device adaptive user interfaces: target device, number of components, nature of the application and number of devices at the same time. From the user tests we concluded that the four parameters have an important impact on the user interface as the obtained rank for all of them is higher than 45%.

In the paper we also propose four different layout templates to be composed automatically based on the application definition: grid layout, menu layout, picture-in-picture layout and horizontal layout. In terms of layout templates, grid is the preferred one by the users. It is a more complex layout compared to the other ones to be automatically composed, so further analysis is needed to evaluate how to organise the components on a grid layout to provide a good experience. Menu (for small screens) and picture-in-picture (for bigger screens) could be the alternative in situations where grid layout does not satisfy the user, since they are almost never the most rejected ones. Finally, horizontal layout is the most rejected by the users, having only sense with a low number of components and in portrait orientation, which means that there could be very specific circumstances where a concrete layout has sense.

These results provide a valuable information in order to create responsive User Interfaces for Web-based multi-device applications driven by media content. However, further research and user tests are needed to collect information from a higher sample of users with different profiles.

## 2.4 Cloud session mainLtenance to synchronise hbbtv applications and home network devices

- **Title:** Cloud session maintenance to synchronise hbbtv applications and home network devices
- **Authors:** Mikel Zorrilla, Iñigo Tamayo, Angel Martin, Igor G Olaizola
- **Conference:** Broadband Multimedia Systems and Broadcasting (BMSB)

- **Publisher:** IEEE
- **Year:** 2013
- **DOI:** http://dx.doi.org/10.1109/BMSB.2013.6621754


**Abstract** - Second screen services encourage TV audience to enjoy new forms of interaction engaging users around TV content as a main thread. This paper describes a standard-based solution for second screen services synchronised with the broadcast content. The user perceives an enhanced broadcast experience enriched with multimedia, textual and social Internet content through multiple devices. The presented end to end solution delegates to a server the cloud session maintenance in order to pair and synchronise HbbTV applications and HTML5-based second screen ones overcoming existing heterogeneous network interfaces barriers of current technological alternatives. The server decides dynamically the behaviour of the different applications regarding the user context, according to his preferences, device features and number of simultaneous views. It also manages the user interaction providing a full synchronised experience thanks to an event-driven mechanism on top of Websockets and AJAX. The paper analyses the performance of the proposed system evaluating the user interaction latency, the concurrency volume of the server and the interdependence, concluding this solution as a suitable approach for broadcast-related second screen services.

**Index Terms** - Multimedia systems and services, DTV and broadband multimedia systems, Multimedia devices, Set-top box and home networking, HbbTV, HTML5, smart devices, broadcast interactivity.

### 2.4.1 Introduction

User experience around television has changed a lot. Second screen paradigm is increasing daily. Many households are already multi-display, using their smartphone, tablet or laptop for sharing with friends through social networks and to interact directly with the programme reaching metadata or additional content. The main tasks that users perform on second screen services include discovery of related contents, finding supplementary information, participation in live shows, shopping of promoted items, and social sharing, faced by different stakeholder actors.

On the one hand, Smart TVs, which have become a commercial success worldwide, provide OS-based heterogeneous platforms to access applications and

services through markets or stores. We can find manufacturer created vertical platforms as Samsung Smart TV or LG Smart TV, or proprietary horizontal approaches as Google TV or Apple TV. These OS-based applications are ready to reach Internet services through the TV but they are not linked to the mainstream, the broadcast content. Users have a discontinuous experience with these platforms, using the device to watch broadcast content or to access applications, ignoring profits of multiple synchronised experiences to capture audience interested in finding further information about TV shows they are watching [LC11].

On the other hand, there are standard approaches involving broadcasters to achieve a synchronised experience exploiting metadata and additional content trough the TV, related to the broadcast content. HbbTV [Mer10] [Mer11] is a pan-european specification published by ETSI[24] to allow broadcasters application signalling and delivery through the carrousel, rendering the application with a CE-HTML based browser in the TV or set-top box bringing a new unexplored market.

Unfortunately, neither HbbTV v.1.1 nor v.1.5 specifications deal with the migration of services to other home devices, such as tablets and smartphones, to achieve a multi-screen user experience synchronised with the broadcast content. These features will hopefully be included in the future HbbTV v.2.0 specification but a solution is needed to overcome this limitation for current deployments.

In this paper we present an end to end solution to synchronise HbbTV applications (both v.1.1 and v.1.5) with mobile devices (e.g. Android or iOS devices) through standard HTML5 applications, using cloud session maintenance in the server side. Thereby the server keeps and manages session variables to identify the different users and their devices, pair them dynamically and adapt the service to the user context, delivering the different target contents to the specific devices.

Section 2.4.2 presents the cloud session maintenance architecture approach. Section 2.4.3 describes performed validation experiments. Section 2.4.4 details the obtained results for the most critical features in terms of low-latency synchronisation regarding the concurrency of the server. Finally, section 2.4.5 shows the conclusions.

[24] Last version: ETSI TS 102 796 v1.2.1 in November 2012

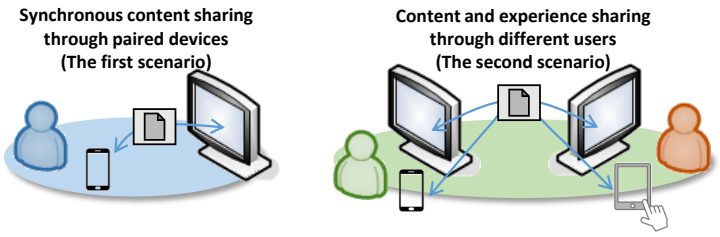


**Figure 2.15:** Scenario diagrams

### 2.4.2 Cloud session maintenance architecture

The different scenarios that compose the scope of the solutions meet additional contents or information around the broadcasted content empowered by means of an HbbTV application on the TV and synchronised HTML5-based applications on the second screen devices (see Figure 2.15). The first scenario involves the synchronous content sharing from the both paired devices on a living room (TV and second screen device), while the second scenario faces contents sharing from any of the involved devices, to friends' TV or mobile through their also paired and synchronised applications even if they are geographically distributed. These scenarios enable the broadcasters to implement Customer Relationship Management (CRM) services with the desire to improve their services and retain their customers [DSN] [DSML+11].

For example, a user, who has already paired a tablet to his Smart TV, is watching a football match on the TV set with his family while he accesses additional contents through the paired tablet application, such as other sport results. Suddenly, a new scored goal of other match is played on the tablet and he wants to share it with other households overlaying the goal replay over the broadcast signal. He can perform this action on the tablet and send it to the TV. The server receives from the tablet the "send it to the TV" event and generates a command for that TV's application in order to show that video (the first scenario). Besides, he shows this goal to a friend of him, which is in another country watching the same match and having the same synchronised experience through a smartphone and a TV (the second scenario).

To address the goals specified in these scenarios our approach defines an architecture based on three main modules (see Figure 2.16): 1) A Node.js[25] web server to manage the cloud session maintenance and the behaviour of the applica-

[25] http://nodejs.org

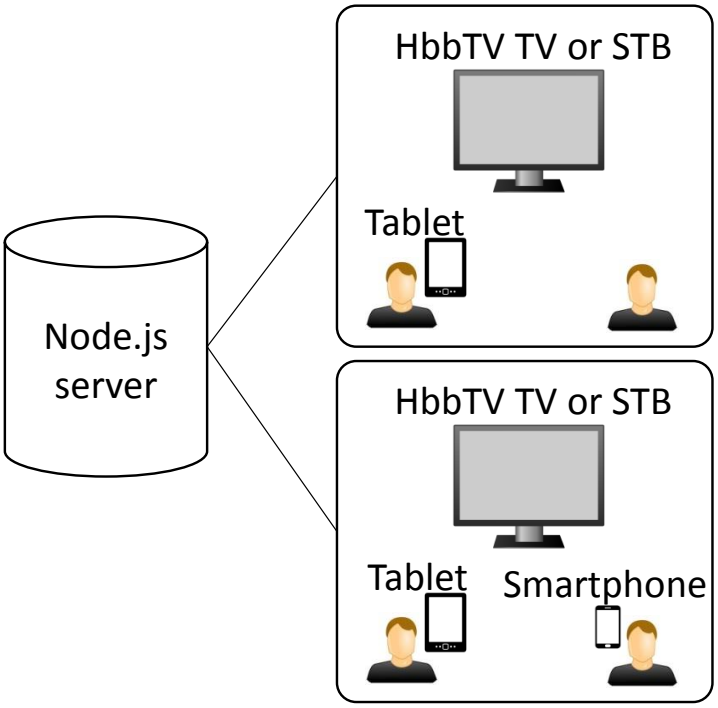


**Figure 2.16:** The different modules of the architecture

tions, 2) HbbTV applications for the Smart TVs and 3) HTML-based applications for the second screen devices (tablets, smartphones, laptops, etc.). Node.js is framework for developing high-performance concurrent programs that do not rely on the mainstream multithreading approach but use asynchronous I/O with an event-driven programming model [TV10]. This enables the server to create low coupling bi-directional communication channels between client applications. It receives user interaction events and generates reaction commands to push unleashed events on other applications.

#### 2.4.2.1 **System Workflow**

The service always starts with an HbbTV application associated to a broadcast content. Each TV application is linked to a unique *session vector* stored in the server which contains contextual information: a) a unique session token, b) information about already synchronised mobile devices and c) user preferences information.

The TV application is not paired with any devices by default, but the user will be able to pair different mobile devices via a web browser. The TV application shows a QR code which contains a URL to the server, including REST-based in-

formation to identify that mobile device with the TV (e.g. the unique session token).

From that moment on, the *session vector* will contain information about the second screen paired with the TV and so the TV application could change it behaviour to adapt to that new context. For example, the HbbTV application shows a menu with some options (one of them is to pair the TV applications with a second screen device) while the user is watching a live event broadcast content. Once a tablet is paired, the TV menu disappears going back to full-screen broadcast signal, meanwhile a secondary display menu is available in the tablet enabling sharing actions of application contents to the TV.

Additionally, the service could be configurable by the user, defining his preferences of what kind of information wants to see in each device, how long overlay it in the TV, etc. All this information will be stored in the *session vector* to influence the parameters of the commands that the server generates.

#### 2.4.2.2 **Server components**

Going into detail, the server has different components to deal with the requirements of the service (see Figure 2.17). On the one hand, it has an HTTP server for the delivery of the HTML-based application to the end-clients; this includes the HbbTV application to the Smart TV and the HTML5 applications to the second screen devices. On the other hand, it has a Websocket module to establish a bi-directional communication channel between the server and the end-devices. This channel is used to receive the user interaction events by the server and to delivery reaction commands to the other synchronised applications. Finally, the server has a *cloud session maintenance* module which takes charge of: maintaining and updating a *session vector* for each HbbTV application; creating the commands and events depending on the multi-device behaviour of the service; and managing the Websocket module and communications.

Concerning latency and data traffic overhead Websockets are the best and more efficient approach for the event communication between the Web server and the clients. Nevertheless, there are some technology restrictions to use them. HbbTV v.1.1 does not have support for Websockets but it is already included in version v.1.5. Regarding the web browsers of the mobile devices, Websockets are included in the roadmap of all of them, but implementations are still missing in

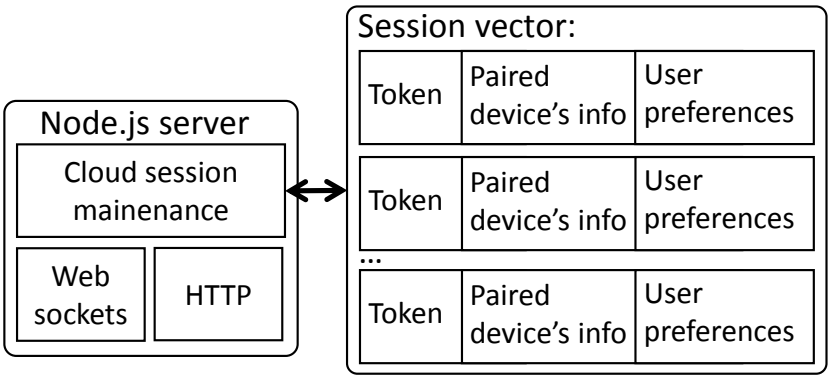


**Figure 2.17:** The components of the server and the session vector structure

some cases. While iOS Safari and Blackberry browser support Websockets, latest versions of Android Browser and Opera Mini do not[26]. The system gets over this limitation using polling mechanisms in the client with AJAX to detect remote activity in the server.

End client HTML-based applications, both TV and second screen device applications, are built over conditional aspects with different behaviour according to: a) presence of second screen devices paired with the TV, b) personal configuration of the user preferences and c) capabilities of the devices in terms of Websockets support, screen size, etc.

#### 2.4.2.3 **Main advantages**

The architecture proposed in this paper brings some significant advantages from other solutions to pair second screen devices to the TV. Most of them are based on local network communication protocols such as UPnP or DLNA. They require configuring the same Internet access point for the household devices. The cloud session maintenance solution pushes to the server the device pairing management. This way it allows the communication between devices connected to Internet no matter how. For example, the user can pair a TV connected to the broadband home Internet connection with a smartphone connected via 3G to Internet. This feature enables multi-user experiences, with the synchronisation and interaction among two people which are not physically together. Finally, it does not require local communication capabilities to the HbbTV devices (UPnP, DLNA, etc.).

[26]March 2013, StatCounter GlobalStats http://caniuse.com/#feat=websockets

#### 2.4.2.4 **Risks**

Nevertheless this scenario has some risks that should be considered, mainly in terms of latency, concurrency and security. Local network based pairing solutions offer low latency characteristics and secure protocols are not mandatory because devices need to be connected to the Home Network (same Internet access point). But cloud-based solutions must keep latency performance in order to grant an appropriate quality of experience to the user even under stress conditions of the platform with massive users performing concurrent requests to the system [TFPK13] [ZWZ+13]. Regarding security, this aspect is not afforded in this paper because it can be faced by any solution on top of the proposed architecture [Far11]. For example, [DLIG12] presents the challenges of building web-based ubiquitous applications over the Webinos[27] platform and facing the privacy and security aspects. By contrast, this paper shows a detailed latency and concurrency analysis of the Node.js-based server for cloud session maintenance services to synchronise HbbTV applications and home network devices as a second screen.

### 2.4.3 **Validation experiments**

In order to measure the latency of the architecture proposed in this paper, two HTML-based applications have been run in a laptop, connected through a local network to the Node.js server. One of the applications acts as the first client, sending the timestamp of the system as a message to the server. The other application is the second client, waiting for a message from the server with the timestamp of the first client to measure the round trip time. This means the second client compares the system timestamp in the moment it received the message and the timestamp received in the message (both applications are in the same laptop so there are not clock synchronising problems). This result is the latency parameter since a user sends an event from one device, until the other device gets awareness, e.g. there are two households in front of a TV, one of them watching interesting additional information related to the content in the TV. Pressing a "Show in the TV" button he just pushes that information to the TV so users can watch it overlaid into the broadcast content.

The values depicted below have been obtained for two scenarios: using Websockets and using AJAX, for those devices which do not support Websockets in

[27] Webinos is an EU FP7 ICT funded project aiming to deliver a platform for web applications across mobile, http://www.webinos.org/

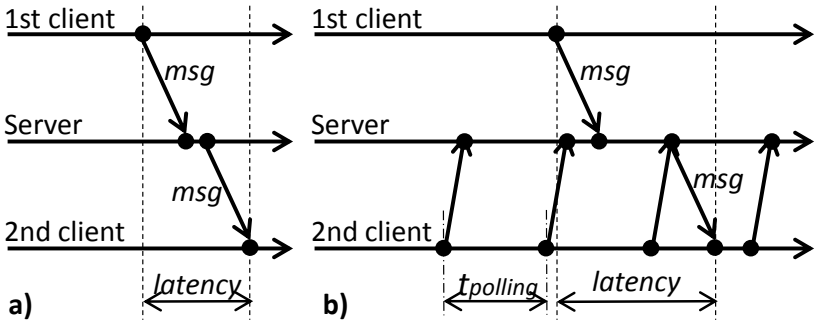


**Figure 2.18:** Latency measure using a) Websockets and b) AJAX

their web browsers.

In the first case, the HTML-based applications open a bi-directional communication channel with the server through Websockets. This way, the first client sends the timestamp to the server, and the server forwards it in a best effort way to the second client (see Figure 2.18a). WebSockets are optimised to reduce communication cost over the Web to a minimum, with a header of only 2 byte compared to the 8 Kbyte header limit for most HTTP messages [GLG11] [LG10].

In the other scenario, the first client opens an AJAX communication to send the timestamp to the server and the second application is recurrently asking through AJAX for a new event to the server. For the experiments, in order to ensure smooth usability with a high Quality of Experience (QoE) we have defined a 100 ms polling interval in the second client to ask for changes in the server. This has a direct impact in latency, adding an extra delay which has to be considered in terms of low-latency needs, and the network load (see Figure 2.18b) [Mob13].

In order to obtain results closer to a public service deployment, benchmarking tools have been employed to simulate massive concurrent requests. This way we assess how its volume affects to the server in terms of latency (see Figure 2.19). We want to establish how many simultaneous users can be managed by the server keeping a minimum value of latency to provide a suitable QoE to the user.

For the simulation of multiple simultaneous AJAX connections we used JMeter[28]. Apache JMeter™ is open source software designed to test the functional behaviour and measure performance of web applications.

Due to unavailability of a benchmarking tool, we have developed a test environment to simulate multiple and simultaneous connection for Websocket

[28] http://jmeter.apache.org/

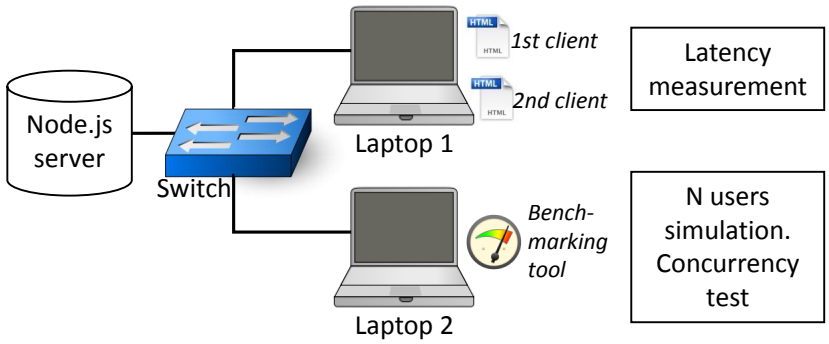


**Figure 2.19:** Architectural diagram for the validation experiments

technology on top of Socket.IO-benchmark[29] and Socket.IO[30] framework, hence we build and adapt them to be used with the Node.js-based server.

Moreover, the same experiments have been done using a server harnessing a variable number of CPU cores to profile the weight of the processing capabilities of the server on the performance. The Node.js-based server has been executed over an x86 64 bit PC with four CPU cores of 2.4 GHz where the OS is Debian 6.0 64bits.

Section 2.4.4 shows the obtained values of the latency regarding the number of concurrent connections to the server and the performance of the server in terms of hardware resources. Moreover, the behaviour of the different communication protocols for these parameters has been analysed.

### 2.4.4 Results

This section details the obtained latency results using the test suite and environment described previously. Using Websockets in a local network, the proposed architecture achieves a latency score of 1,97 ms since the user presses a button on one device, until he notices the reaction of the service in the other device. These tests have been done using a single CPU core on the server and a four-core CPU. If we analyse the performance for a variable number of users, a single user using two devices obtain the same latency results mentioned before. For a low number of concurrent connections with the server the latency values are from 1,9 to 4 ms. In this cases, the OS multi-core management of the server increases slightly the latency (see Figure 2.20).

[29] https://github.com/michetti/socket.io-benchmark
[30] http://socket.io/

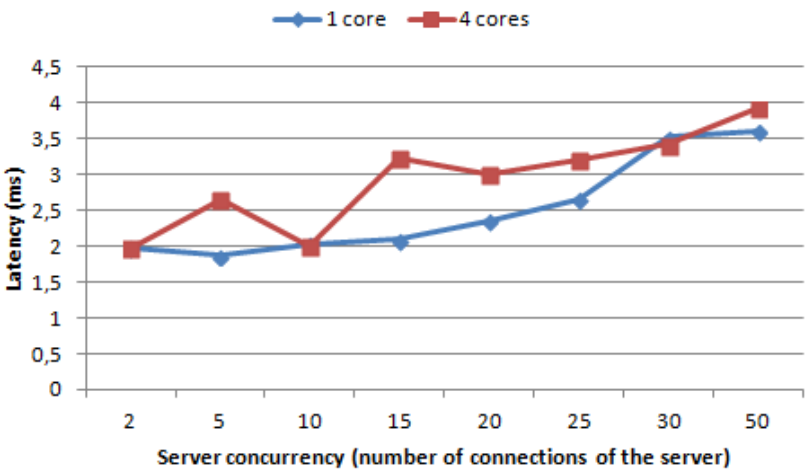


**Figure 2.20:** Latency of the system with a low number of concurrent connections using Websockets

Testing how the latency behaves when increasing the number of devices simultaneously sending events to the server, the CPU of the server is the bottleneck. While there is a stable zone with an appropriate latency threshold with a server of one and four cores, increasing the number of concurrent connections both react different. Once the CPU is overloaded, the latency increases exponentially. Figure 2.21. shows how the stable zone still continuous for a four core server for 3000 simultaneous connections while the one core server is drastically overloaded.

Same tests have been performed with AJAX asynchronous connections between the clients and the server for those clients which do not implement Websockets. As seen in Figure 2.18 the client architecture to notice activity in the server needs recurring connections from the second client to the server. The tests have been done with a 100 ms slot from one connection to the other so it adds 50 ms of latency of average since all the latency values have been calculated with the average of 100 measures.

For better latency values we could reduce the 100 ms slot but a trade-off is needed between the latency and the overhead of the network and server in terms of applications design. Figure 2.22 shows the latency using AJAX for a low number of simultaneous connections to the server with a one-core and a four-core server.

The minimum latency value obtained is 54,47 ms, which matches 4,47 ms to the system architecture since 50 ms are added in the client applications design. Comparing Websockets and AJAX, there is an increase of 226% using AJAX, be-

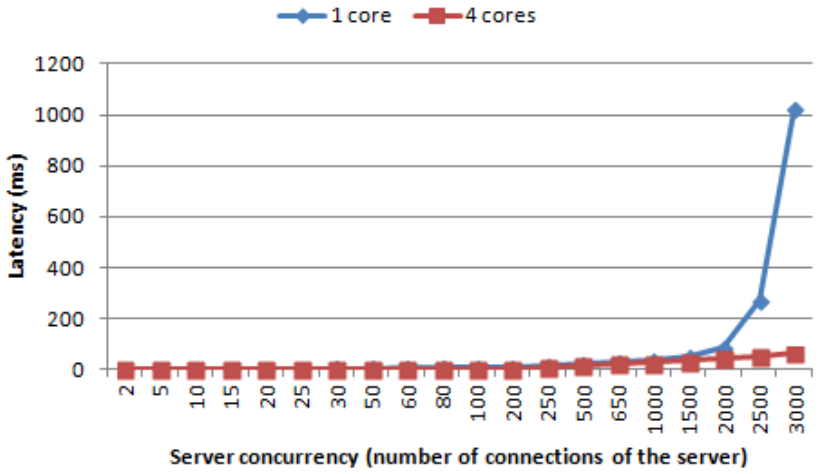


**Figure 2.21:** Latency of the system with a high number of concurrent connections using Websockets

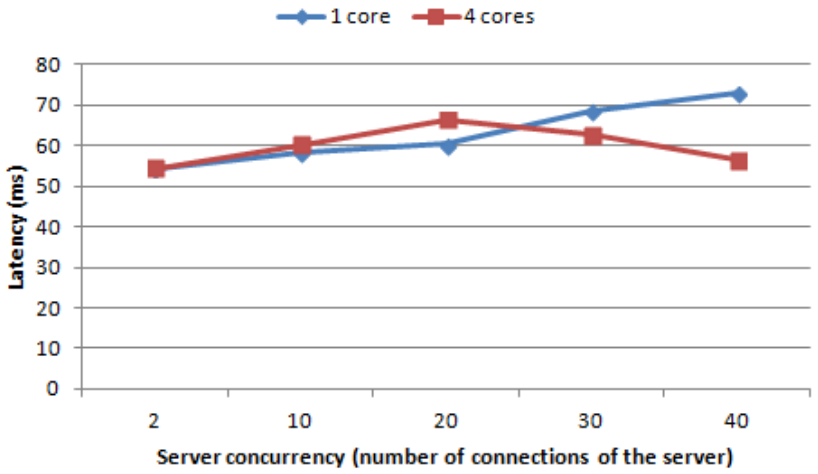


**Figure 2.22:** Latency of the system with a low number of concurrent connections using AJAX

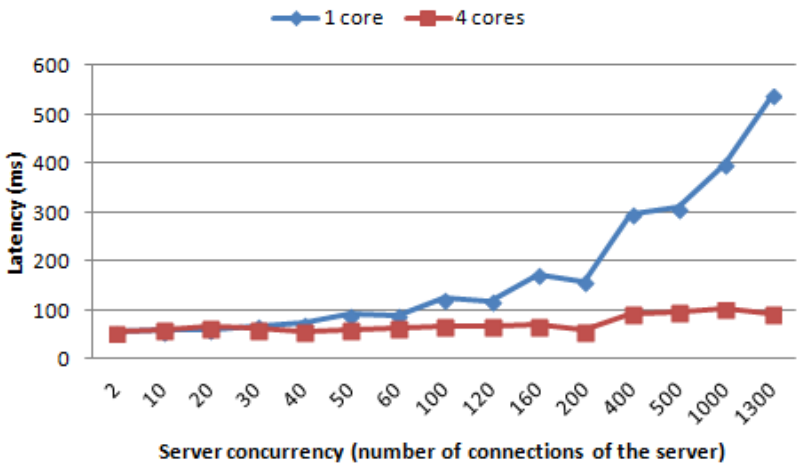


**Figure 2.23:** Latency of the system with a high number of concurrent connections using AJAX

cause it needs to open the communication channel each time. Using Websockets the applications open the socket session at the beginning and then just send the messages. But even with AJAX, the system latency is very suitable for a wide range of applications under these conditions.

Using AJAX has a higher impact in the server CPU so it accepts less simultaneous connections inside the stable zone latency. Figure 2.23 shows the latency for a number of AJAX connections from 2 to 1300.

### 2.4.5 Conclusions

This paper introduces a cloud session maintenance architecture to synchronise HTML-based connected TV applications (HbbTV) and home network devices such as smartphone or tablets via their web browsers and provide concurrent second screen services.

Instead of turning on local network communication protocols to pair the second screen devices with the TV set, we define a server-based architecture that pushes to the cloud all the pairing responsibilities. It removes local area network constraints for the home multimedia devices such as using local network communication protocols (UPnP, DLNA, etc.) and share the same access point for household devices, and extends user synchronised experiences to Internet connected applications.

Moreover, different communication protocols have been analysed to send interaction events between the different clients and the server. On the one hand, we

employ Websockets as the most efficient bi-directional communication channel. But for some devices, such as HbbTV v.1.1 equipment, some smartphones and tablets that do not support Websockets, an alternative architecture is proposed on top of AJAX.

The validation experiments have been described to test the proposed architecture in terms of latency and to parameterise the server performance for a high volume of users accessing concurrently. All the experiments have been carried out over a Node.js server with a variable number of CPU cores and connecting all the devices through a local network.

The results of the validation experiments conclude that the latency of the proposed architecture is suitable for most of the second screen services that can be envisaged over this solution. Websockets provide a much better latency mainly related to the provided bi-directional communication channel capability enabling the server to send events asynchronously to the clients. Nevertheless, an AJAX-based solution could be also adequate for a wide range of less demanding second screen applications.

Incrementing the number of simultaneous connections that are exchanging information with the server, there is a *stable zone* where the latency value remains quality threshold. This shows that the server architecture is ready to perform a multi-device ecosystem of many users interacting with synchronised contents concurrently. Nevertheless, there is an inflexion point where more users accessing simultaneously affect drastically to latency. This point is closely related to two main factors: 1) the communication protocol used between the clients and the server, being Websockets the most efficient solution dealing with a higher number of simultaneous connections without a negative impact on the latency and 2) the available CPU cores on the Node.js-based server. Using a single core CPU server the inflexion point rises with a limited number of concurrent connections while with four cores the number of connections accepted on the *stable zone* increase. These results demonstrate that the architecture is suitable for a high number of users using simultaneously the service. So, in order to manage adequately to a volume of concurrent requests for the latency *stable zone*, the server just must dimension CPU available hardware.

To conclude this paper provides a suitable end to end solution based on a cloud session maintenance architecture that allows synchronised second screen services with Connected TVs and mobile devices (smartphones and tablets). This architecture performs a solution over already commercialised devices such as

HbbTV compatible Connected TVs (v. 1.1 or v.1.5) and Android, iOS, Windows, Blackberry, etc. smartphones and tablets with a standard mobile web browser.

## 2.5 Reaching devices around an HbbTV television

- **Title:** Reaching devices around an HbbTV television devices
- **Authors:** Mikel Zorrilla, Angel Martin, Iñigo Tamayo, Sean O'Halpin, Dominique Hazael-Massieux
- **Conference:** Broadband Multimedia Systems and Broadcasting (BMSB)
- **Publisher:** IEEE
- **Year:** 2014
- **DOI:** http://dx.doi.org/10.1109/BMSB.2014.6873499

**Abstract** - HbbTV takes advantage of the opportunity to expand the broadcast experience exploiting the common media content. However, the time to reach the audience in a different way has come. Aware of the privileged position of the TV in the living room, manufacturers and marketplace app developers have fostered their own bunch of solutions to integrate the big TV display with the mobile ones, to consume broadband media but ignoring the broadcast traction potential. One major challenge of all these approaches is the resource discovery and association step, where different strategies have been employed. A TV content-centric approach opens new possibilities. First, the possibility to enhance the offer services scheduled on a time basis according to the broadcast signalling. Second, the awareness of a common media been played at the same temporal and spatial environment can support the discovery and association of surrounding handheld devices. This paper analyses the capacity of common visual and acoustic environmental patterns to build enhanced discovery and association protocols, concluding a multi-step combined solution as a suitable approach for broadcast-related second screen services.

**Index Terms** - Multimedia systems, Digital multimedia broadcasting, Pervasive computing, Ubiquitous computing, Context-aware services.

### 2.5.1 Introduction

The new audience habits when enjoying the TV are changing its prominence introducing new entertainment displays in the living room, such as smartphones and tablets, all being connected to the Internet. This means users watching the

TV at the same time as interacting with their handhelds. Thus, the TV is not always the foreground device but one more, combined with others. However, in this new multi-screen paradigm the TV has still the best position at the home area to attract all the surrounding devices providing enhanced experiences through all the displays. Moreover, it is often the only device that gathers connectivity to the broadcast signal and to the Internet at the same time.

Users are ready to new ways of enjoying TV linear media content with seamless multi-screen media services. This includes the migration of visual components from one device to another, creation of multiple views through different devices simultaneously and sharing social experiences with a friend or a community.

However, these services bring distributed computing and pervasive computing scientific challenges such as the service and device discovery, adaptation and self-organisation to dynamic scenarios, communication and synchronisation between devices, etc. These challenges remain on a particular way on this new home environment around a TV.

Connected TVs and Smart TVs are the bet from TV manufacturers and OS-based vertical approaches to face usage shifting and extend interaction possibilities. Nevertheless the full proprietary based ecosystems of multimedia services have created fragmented markets introducing interoperability problems that hurdle multi-screen experiences. Moreover, the broadcasted linear content is often not included in the media services of these proprietary solutions, reducing the TV capabilities only to the Internet connection.

Other stakeholders in the TV, producers and broadcasters, need their own standard technological solution. The enabling technology that opens a wide spectrum of possibilities for the broadcasters to reach audience is HbbTV [Mer11]. It enhances the linear TV programming through non linear Internet-based content, providing enriched services over the mainstream TV content.

Current HbbTV 1.5 specification enables access to media services through the TV and its remote control but it does not cover the multi-screen environments. Thus, the research activity around this standard focuses in including mechanisms to communicate the TV with other devices around. To overcome this limitation there are a wide range of market solutions and different research approaches for the discovery and association of devices. However, there are still open issues that must be covered to apply these solutions to an HbbTV such as building multi-domain network connectivity of surrounding devices and delivering broadcast content centric services.

This paper focuses on the resource discovery and association procedures in order to be able to provide multi-screen media services around an HbbTV television.

### 2.5.2 State of the art

This section overviews the state of the art from three different point of views; a) the market solutions for device and service discovery and association, b) the discovery and association challenges in distributed and pervasive computing, and c) the translation of the general scientific challenges to a TV centric home environment.

#### 2.5.2.1 Market approaches

The industry solutions for resource discovery and association are several such as UPnP, Bluetooth, Wi-Fi Direct or Zigbee. However, some of them are based over hardware connectivity interfaces not present on all the devices (including TVs), while others require a common network interface that could not be enabled through all the devices at the same time (e.g. handheld devices are often in a home environment connected through 3G, out of the home gateway).

Concerning zero-configuration networking standards, the most widespread protocols are: DNS-based service discovery (DNS-SD), multicast DNS (mDNS), UPnP Simple Service Discovery Protocol (SSDP), Service Location Protocol (SLP), W3C Web Intents [LNIF13], Bluetooth Service Discovery Protocol (SDP), Web Services Dynamic Discovery (WS-Discovery) [JMS05], Discovery and Launch Protocol Specification (DIAL).

This research field has a lot of activity and we can find different software frameworks that face these topics such as: XMPP Serverless Messaging, WS-Discovery Implementation (Apache Foundation) and Google Cast, exploiting the previous mDNS, WS-Discovery and DIAL protocols respectively; Webinos Discovery API[31], considering additional application areas such as in-car displays; AllJoyn, exploiting the SoC potential for specific architectures; etc. Other working groups like IETF Homenet aims to simply deploy home networks based on device density explosion linked to the IPv6 technology.

[31] http://dev.webinos.org/deliverables/wp3/Deliverable34/servicediscovery.html

This well-stocked protocol landscape provides a set of solutions tailored for multi-screen applications and broadband services. They open new browsing and interaction possibilities for marketplace apps developers and enhance the broadband and local media sharing integrating TV and handhelds. Nevertheless, not only none of them brings a significant and direct added value for the broadcaster, but also they do not take benefit of the broadcast signalling and media nature.

#### 2.5.2.2 **Discovery and association challenges in distributed and pervasive computing**

In the field of pervasive computing environments, arisen challenges in terms of resource discovery and association have been studied for long time [ZMN05] and defined solutions are still open to new approaches tracking new paradigms and embracing incoming technologies.

Once the most representative solutions have been depicted, it is necessary to understand the different protocol features to match the proper discovery and association technology with the HbbTV multi-screen scenario needs. To this end, it is required to take into account the communication overhead and to consider the performance degradation trade-off. A service discovery protocol must be designed to minimise administrative overhead and increase usability.

In this line, but going deeper, the classification of the resource discovery and association protocols will support later decisions to build a suitable mechanism:

**Presence Discovery**. Its objective is to find the specific address though the active network interfaces of each surrounding device. In this case, the options are: unicast, requires factor preset or manual set up to obtain environment awareness; multicast, try to provide a fully automatic solution reaching any device by flooding the network with UDP messages; broadcast, with the same principle but constrained to a single hop.

**Naming Service**. Its aim is to provide the available service description. In this case, there are two possibilities depending on the preliminar mutual knowledge: preset, including service names, parameters and structure; template, providing a common format. The preset option involves a minimum overhead and transparent discovery but reduces significantly the expressiveness and expandability for new services, while the template one will require often human intervention.

**Service Availability**. Its target is to report the list of services in a device. The alternatives are: notification, only listening devices get awareness; querying, a device ask for features and capabilities to a specific one when it wants to interact.

The periodic notification introduces traffic overhead so the continuous polling is not the best option. Service Publication. Its purpose is to keep the list of the available services. In this case, the options involves: a centralised publication board, which must be hosted in a stable device in terms of availability; peer publication, where each device share just their services. Obviously, the first one enables a wider integration for applications where all the devices participate, not just peers.

**Discovery Scope**. Its goal is to define service discovery sessions gathering the appropriate data including: network topology, this information is based on domain belonging; user role, needing of user authentication; high-level context, including temporal, spatial and user activity. The network information is a key factor in environments with heterogeneous connection interfaces. Anyway, when trying to launch a seamless experience just in time, it is important to avoid explicit user authentication because it usually brings additional user interaction that restrains the usage willingness.

#### 2.5.2.3 Translation of discovery and association challenges to the home environment around the TV

This section tackles the adoption of the distributed and pervasive computing discovery solutions to multi-screen scenarios with the TV as the main device on the home environment.

Regarding *presence discovery*, there are two relevant issues to keep in mind. First, while the TV is mostly connected to Internet through a home gateway (WiFi or Ethernet), mobile devices are usually shifting from the home gateway (WiFi) to 3G. In this case, a broadcast mechanism will be constrained to a specific network area or domain, not reaching every device.

Second, only HTTP services are supported in HbbTV devices and most of the multicast discovery protocols are over UDP (UPnP, etc.). Web Intents could be a solution to exploit in the browser other resources of the TV. The W3C is also addressing this challenge with The Network Information API working draft[32] and Network Service Discovery working draft[33].

Discarding unicast option in order to foster transparent solutions, it is necessary to create new strategies that exploit media broadcasting nature combined with the surrounding presence awareness step of the discovery protocol.

[32]W3C Working Draft 29 November 2012 http://www.w3.org/TR/netinfo-api/
[33]W3C Working Draft 20 February 2014 http://www.w3.org/TR/discovery-api/

Concerning *naming service*, it is mandatory to avoid human setup. This point would enhance the HbbTV technology potential but requires processes for specification expansion through releases, while meeting its business model.

In terms of *service availability* and *service publication*, the notification option fits better with the centralised publication board, achieving a major inclusive scope embracing all living room devices participation. Moreover, notification mechanisms enable event-based services that fits perfect in the broadcast business model.

Regarding the *discovery scope*, some considerations apply; From a network topology point of view, there are different discovery protocol stacks over a LAN, but they need all the devices to be connected to the same gateway and are mostly based on UDP. However, these technologies cannot be deployed on top of pure HTML devices like HbbTVs. From a user role perspective, the authentication is a key element which is a big challenge by itself in multi-user multi-device environments. Webinos creates trustworthy zones where the user can share devices with a community. From a context point of view, there are temporal and spatial approaches for the discovery scope. The main spatial scope is the physical location: using sound or light patterns, QR code scanning, bumping devices synchronously, etc.

### 2.5.3 **Proposed solutions**

This section provides different solutions to overcome the open issues introduced in the previous section, designing multiple approaches that have been landed by the implementation of spatial discovery scope solutions regarding the physical location, where the different devices are all around the TV.

All these solutions are designed to be implemented over current TVs or Set-top boxes with HbbTV v1.1 or v1.5. The TV will receive a related service from the broadcast carrousel and the HbbTV application must discover the devices around the TV. An association will be done between the second screen device applications and the HbbTV application.

Event-driven server based approaches enable the communication between the associated devices for the visual component migration, multi-screen views and social experiences between users. On [ZTMO13] an event-driven solution to synchronise HbbTV applications and second screen devices is proposed. [Zie13] presents a standards-based framework that enables bi-directional communication between HbbTV applications and second screen devices. These event-driven

solutions need a previous association step. Often these association mechanisms are based on a common server session established by means of a common HTTP URL accessed by a set of devices. This is done discovering an HbbTV application in the TV, and sharing the URL with the second screen device and a unique ID created by the HbbTV application that allows the server to associate a unique HbbTV applications with a unique HTML-based mobile application. The following solutions address how to exchange this URL and the unique ID amongst the devices in a friendly way for the user.

#### 2.5.3.1 **Visual solutions**

A visual approach is a main basic solution to take advantage of the visual information that the user is watching in the TV. HbbTV can show in the application a URL with the related service and a unique ID for that TV app session. The user introduces manually the URL in his mobile device. The ID could be included in a second step, inside the mobile application as a PIN or code it in a REST URL.

A usability improvement for the user is to use QR codes. The HbbTV application generates a unique QR code where the URL and the ID are coded inside. The user scans the QR code with the mobile device and the second screen HTML application starts already associated with the TV session. This is the current association approach for [ZTMO13] and [Zie13].

In this case, the mobile device must have a camera and a native application to scan the QR code and open a web browser with the URL and the ID information on the TV. These requirements are very reasonable for current smartphones and tablets but require a tedious process for the user, where first has to navigate inside the HbbTV application with the remote control until the QR code is shown, and then open a QR scanning application on the mobile device. Depending the distance between the user and the TV, the size of the QR code, the data amount in the QR code and the camera of the mobile device, the experience of the user changes a lot. Next section provides the experiments carried out to characterise these parameters.

#### 2.5.3.2 **Sound patterns**

The use of sound patterns is a more transparent approach to exchange information between two devices that are physically close. The broadcaster is able to overlay out of band sound patterns over the TV programme. A URL could be

coded inside that sound pattern, but if it is done in the broadcast signal, it is not possible to generate a unique ID for each TV.

The HbbTV applications are also able to generate a sound pattern, unique for each application but there are limitations to overlay an extra audio to the broadcast signal in HbbTV. HbbTV v1.1 and v1.5 have a *video* object and the specification[34] says that "Attempting to present broadband delivered video using the AV Control object may result in suspension of access to broadcast resources, including (...) Broadcast video presentation being stopped". This clause applies to terminals which do not have the HW capability to present broadband delivered video at the same time as de-multiplexing MPEG-2 sections from the broadcast. Indeed, most of the HbbTV devices.

This interruption of the broadcasted content to play the application generated audio is unpleasant to the user. Next section presents experiments with different TVs and STBes to assess the QoE related to measures of the interruption time when switching and managing the *video* object in HbbTV.

#### 2.5.3.3 **Bump based solutions**

Bumping two devices is a solution that has been used to pair mobile devices. Bump[35] was an application for Android and iOS to exchange images and videos from one device to another with a simple bump between devices. It was also possible to send content from a mobile device to a PC pressing the space bar of the computer and at the same time the mobile phone is bumped. In September of 2013 Bump Company announced that Bump was joining Google to continue exploring the user interaction with mobile devices[36].

This association paradigm can be transferred to a TV centric home environment. When the HbbTV applications are loaded, instead of showing the typical "press red button" message, shows a "bump based message available". The user shakes the mobile phone while pressing a button in the remote control and from this moment on the association is done.

The presented bump based solution is based on an event-driven server that receives requests from different devices and associates them according to the *timestamp* of the request and some additional information of the request (such as the location). The accuracy on the *timestamp* measure on both devices, and

[34] ETSI TS 102 796 v1.2.1 (2012-11), known as HbbTV 1.5
[35] Bump: https://bu.mp
[36] http://blog.bu.mp/post/61411611006/bump-google

a precise location of each device is critical in order to avoid false positives. The next section shows experiments in order to evaluate the latency of the request from different networks and the accuracy to associate them in a TV centric home environment.

Due to the bump-based solutions needs the user to press a button on the remote control, to develop a more robust approach the server can operate a button-hopping mechanism to spread the interaction spectrum. Thus, exploiting all the standard buttons available on the remote control. The mobile device will indicate the user which button to press while shaking the mobile so the event-driven server can conclude unambiguously between different request coming in the same moment and from a similar location.

The bump based approach is much more user friendly since the user directly starts the experience associating the devices, and could be used with multiple second screen devices with a single TV at the same time.

#### 2.5.3.4 **Three step solution**

The previously mentioned solutions could be used combined providing a new solution to discover and associate an HbbTV television and a second screen device. This proposed combined solution follows a three step cloud-based distributed approach. In a first step, the server collects all the bump based requests received from different devices, analyses the data related with the request, and finds the correct associations between requests.

As explained before, the main feature to associate two devices is the timestamp of the requests together with the additional information (location of the devices and the button-hopping mechanism). If the first step is enough to associate the devices exceeding a predetermined threshold of success, the process ends here.

However, with millions of user discovering and associating devices at the same time, the bumped-based solution could be complemented with other two steps as a scalable verification method to minimise the uncertainty.

In the second step, the server will activate a verification protocol based on sound patterns. The HbbTV application will play an audio pattern and will expect that the presumably associated second screen device will listen to that pattern. This verification protocols are executed seamlessly to the user.

Finally, as a last and secure solution, a second verification task could be done as a third step if the first two steps are not enough. The server can send a visual

information (such as a PIN number or a QR code) to the HbbTV application to display it in the TV and wait until the user introduces the information in the mobile device (writing a PIN of scanning a QR code). This third step is the most explicit for the user but provides security and certainty.

### 2.5.4 Experiments and validation

This section presents the experiments outlined in the previous section to define technological thresholds for suitable environments. On the one hand, some experiments have been performed to assess QR code scanning accuracy between a TV and a mobile device. On the other hand, some tests about experience interruption to play an application generated audio pattern in an HbbTV device have been done. Finally, other experiments have been done to evaluate the timestamp based accuracy of an HTTP event-driven approach.

#### 2.5.4.1 QR code scanning

Four parameters have been identified as the main factors involved in the QR code scanning process:

- **QR code size**: The size of the QR code rendered in the HbbTV application in the TV. Two different sizes have been used in the experiments: a 264x264 px and a 396x396 px in a 1280x720 px HbbTV application and a 40 inches TV.

- **Character length**: The number of characters coded in the QR code. We used a QR code with a 75 characters URL and a shorten URL of 21 characters using bit.ly[37].

- **Scanning distance**: Distance from the mobile device to the TV. Different distances have been tested in the experiments: 2 m, 3 m and 4 m.

- **Capabilities of the camera of the mobile**: Two different tablets have been used. A Nexus 10 and an Asus Eee Pad Transformer TF101 Tablet. Both have 5 MP cameras but different optics and features.

[37] https://bitly.com/

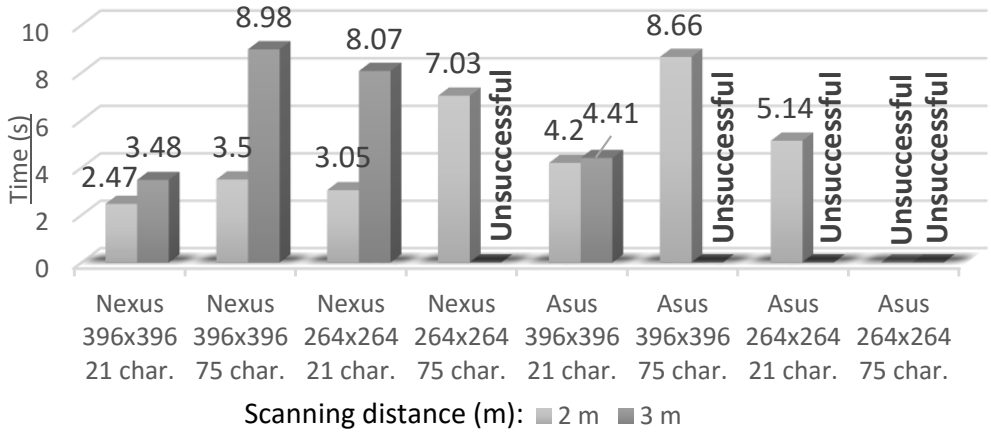


**Figure 2.24:** The QR code scanning time (s) for the successful processes depending the scanning distance, the device, the length of the QR code and its size

The QR code has been rendered by an Engel EN2000 set-top box, inside an HbbTV application. The Barcode Scanner[38] application has been used in both mobile devices (Nexus 10 and Asus). The illuminance in the room was 84 lux. As the outcome of the experiment, the process has been evaluated as successful or unsuccessful and the time for the scanning process has been measured.

Figure 2.24 presents the QR code scanning time in seconds for the successful processes depending the scanning distance, the device, the length of the QR code and its size. As a conclusion, the longer distance, the higher the probabilities of an unsuccessful process it is, and the higher the scanning time for the successful processes. While almost in all the conditions the processes have been successful with a distance of 2 m, there were not successful processes with a distance of 4 m.

The scanning time was faster and with more success index with a QR code of 21 characters than with 75 characters, and for a size of 396x396 px instead of 264x264 px. The experiments conclude that a larger distance of scanning can be address while the QR code size is bigger and the URL coded in the QR is shorter. On the other hand, the quality of the camera of the mobile device has a big impact in the scanning process. In this case the Nexus 10 has a better camera than the Asus under the same conditions in all the cases.

[38] https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=en

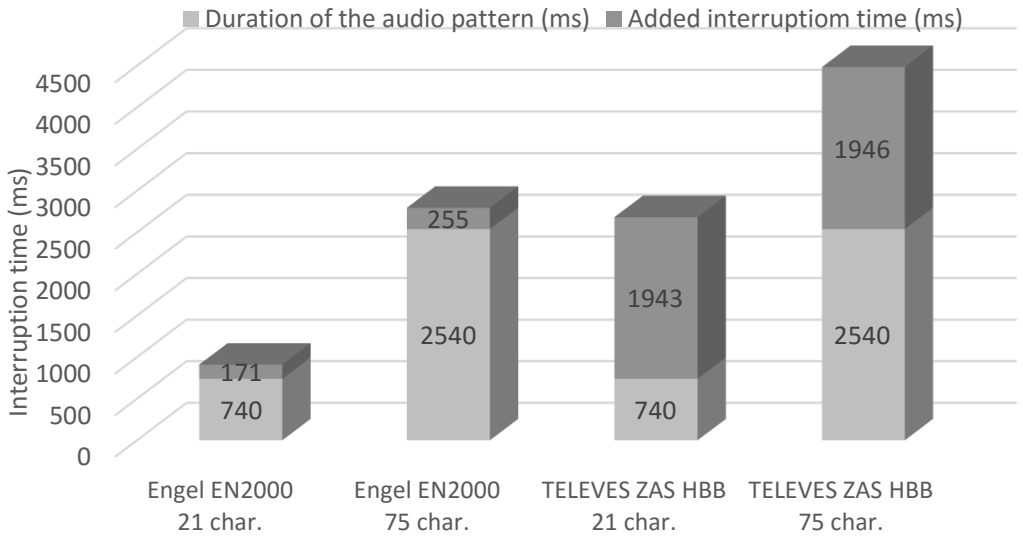


**Figure 2.25:** Interruption time in EN2000 and zAs Hbb Set-top Boxes measured for a 21 and 75 characters long URL coded in an audio.

#### 2.5.4.2 Playing a broadband audio in HbbTV

Due to the limitation of HbbTV specification to overlap a broadband audio to the broadcast signal, these experiments measure the interruption time since the broadcast signal is cut in the TV, until the broadcasted content is back after played the sound pattern, recognisable by the second screen device.

We used Minimodem[39] to generate the audio AFSK modulated patterns for both coded URLs presented in the previous section (21 and 75 characters long). For the long URL, we obtained a WAV file of 2540 ms, and for the short URL, a WAV with a duration of 740 ms. After some AFSK tests, the best configuration is 300 baud rate, 1700Hz for mark frequency and 1060Hz for space frequency guaranteeing successful data transmission on a noisy environment (58dB of background sound).

We created an HbbTV application[40] to interrupt the broadcasted video as less as possible to play the sound pattern, and measured the interruption time. We have confirmed with a Minimodem capture console that the URL was well transmitted on that acoustic environment.

Figure 2.25 presents the obtained interruption time for 21 and 75 characters long URLs coded in an audio pattern with two different HbbTV Set-top Boxes: ENGEL EN2000, and Televes zAs Hbb.

[39] http://www.whence.com/minimodem/
[40] https://github.com/itamayo/Fsk_test_4_hbbtv

The results show that the interruption time goes from 911 to 4486 ms. While the Engel introduces only an added interruption of around 200 ms, the Televes STB adds almost 2 seconds. In all the cases, the experience for the user is acceptable, above all as verification process as proposed in the *three step solution*.

#### 2.5.4.3 **Timestamp accuracy in a TV centric home environment**

We have done four different experiments to measure the timestamp accuracy for a *bump-based* solution of discovery and association of a second screen device around a TV.

For these experiments we have created an HbbTV application for the TV and a HTML5 application for the mobile device. Both applications send a message when the user interacts.

From a user perspective, the user interacts with both devices at the same time (e.g. pressing a button on the remote control while shakes the mobile device). For each of the four experiments we have done 50 measures and got the average to compensate the random precision of users pressing two buttons at the same time. The event-driven server has been published in an Amazon Web Services server.

In the first experiment, we measured the time difference comparing the server reception timestamps. Both devices were in the same local network area. The average of the difference of the events received in the server has been 150 ms.

In the second experiment, we also compared the server side reception timestamp for both events. In this case the Set-top Box was connected to a home gateway while the mobile device was using an HSDPA phone network. In this case, the average of the difference between both events has been 1584 ms. It increases due the heterogeneous networks and the different trace to reach a same server.

In the third experiment, we captured in each client application the interaction timestamp and send it to the server. The server compares the difference between the two timestamps received. In this case, both devices were in the same LAN. The devices were configured to automatically synchronise their clocks and the LAN provides a common clock for both devices. The average difference between the received events is 528 ms.

In the last experiment, we also captured the timestamp in the client side and send it to the server, but in this case the STB was connected to the home gateway, while the mobile device was using an HSDPA phone network. Both devices

were also configured to automatically synchronise their clocks through the network but they were in different networks. The average difference between the timestamps has been 3315 ms.

Using client-side timestamps we obtain worst results comparing to the server-side timestamps. This is due the inaccurate clock synchronisation of the different devices, more evident when they are in different networks. We could force user to set up a common NTP server on all his devices to synchronise the clocks of all the devices involved in the process, but this does not seem realistic.

The first and the second experiments seem to provide accurate scenarios, where we have a server on the Internet and in the worst case two devices at home connected through different networks. The latency jitter ranges from 150 to 1584 ms. The *bumped-based* approach can be strengthen with additional information in the request messages to overcome the association uncertainty for requests closer than 1584 ms. Moreover, the *three step solution* improves the timestamp mechanism with additional information and provides extra verification steps.

### 2.5.5 **Conclusion**

This paper proposes different solutions to discover and associate a second screen device around an HbbTV television. Three different discovery and association solutions are presented landed by the implementation of spatial discovery scope approaches regarding the physical co-location.

On the one hand, *visual solutions* are proposed to deliver a URL and a unique ID generated from an HbbTV application from the television to a mobile device, represented by widespread QR code technique. It requires a second screen device with a camera and a native application for the QR code scanning. Some experiments are presented to evaluate the main factors involved in the QR code scanning process from a mobile device to a TV.

On the other hand, *sound patterns* are proposed for the delivery of the association information from the TV to the second screen device seamlessly. Due to the necessity to generate a unique ID for each TV, the sound pattern should be generated in the TV application itself and the paper analyses the current problems to overlap an HTML-based audio to the broadcasted signal in HbbTV. Experiments are presented to measure the interruption time of the mainstream media since the broadcast signal goes to background, until it is back to foreground after played a sound pattern recognisable by the mobile device.

As the third approach, a *bumped-based solution* is presented in the paper to discover and associate the second screen devices around an HbbTV television. It is based on a synchronous event-driven server that receives the different requests and associates them depending on the reception timestamp and some additional information (the location of the devices and a button-hopping mechanism). The paper presents some experiments measuring the delay between the different requests received by the server through various networks.

Finally, a *three step solution* based on the combination of the previous approaches is presented as the most user friendly and robust. All the experiments confirm the viability of the discovery and association approach as a suitable solution for the discovery and association of a mobile device around an HbbTV television in broadcast-related second screen services.

CHAPTER

# 3

# Two-way complementarity of computing capabilities in multi-device media services

## 3.1 Context

The consumption habits of the end-users with media services have changed considerably. Users want to consume audiovisual content anytime, at any device, and often using more than one device simultaneously. Device fragmentation in the home environment, where Smart TVs live together with laptops, tablets and smartphones, is increasing more and more with the irruption of smart watches and bracelets, as well as the proliferation of screens to consume media in other environments such as the connected car. Device heterogeneity causes consequently heterogeneous performance capabilities on the clients to consume media services. Powerful devices such as PCs and laptops, coexist with increasingly powerful but still thin ones such as TVs, set-top boxes and mobiles.

Media services themselves have also evolved a lot in recent years. On the one hand, the media goes further than the traditional audiovisual content, enhanced with computer generated content and immersive experiences among others. On the other hand, the social media paradigm has led to a significant rise in the volume of user generated content managed by social networks with millions of users accessing services. Service providers aim to engage audience, eager

for contents, by boosting the media relevance. To this end, a deeper automatic tagging enables better matching of user interests with the content database and reveals underlying connections between items, such as applying face detection mechanisms or content-based indexing to find related videos. Image analysis algorithms empower automatic retrieval of salience features but they also involve computing-intensive functions. Therefore, the processing requirements grow substantially when all the media items comprising the social network database are analysed. Here, on the one hand big data challenges arise when social services have continuously increasing databases, while on the other hand more and more processing resources are required to analyse all the content.

To address these requirements, this section presents a solution that enables a two-way complement for computing capabilities enhancement, driven by the favourable and increasingly improved communication conditions following the trend of 5G networks.

Section 3.2 provides a solution where cloud server computing can complement in real-time thin devices when they require demanding computing capabilities, such as rendering computer generated personalised and interactive 3D content as part of the media service. A system architecture called 3DMaaS is proposed to provide complementary rendering capabilities to thin devices, adding to their own capabilities the chance to push to the cloud complex 3D rendering tasks while dealing with a hybrid local-remote rendering.

Finally, Sections 3.3 and 3.4 propose a solution where thin devices can complement a cloud server for delay-tolerant computing tasks, since client devices are often idle or underperformed while watching videos and can contribute with part of their resources to perform atomic tasks of image analysis to enhance the media experience, such as creating automatic tagging, and save costs in the could resources.

While Section 3.3 provides a distributed solution to exploit the idle CPU resources of the mobile devices, in Section 3.4 a enhanced system called SaW is proposed as a solution for media analysis of the content collection in a social service exploiting both CPU and GPU resources of the client devices. Drawing inspiration from volunteer computing initiatives for big data, this solution presents a pure Web-based distributed solution. SaW is deployed on top of the user appliances of a social media community, including the hardware-accelerated features for suitable devices. Thereby, the service provider gains the immense processing ability of its big social community to perform independent background

hardware-accelerated image processing tasks. To achieve it, these queued tasks are embedded to the different social media services accessed by the users. On the one hand, a pure Web-based architectural design of the SaW concept is presented, enabling an interoperable solution. On the other hand, the a proof-of-concept implementation of SaW using WebGL and WebCL technologies is provided, in order to evaluate the SaW approach supported by experimental results and an analysis of the performance based on a previous model extending it by terms of GPU usage.

## 3.2 HTML5-based system for interoperable 3D digital home applications

- **Title:** HTML5-based system for interoperable 3D digital home applications
- **Authors:** Mikel Zorrilla, Angel Martin, Jairo R. Sanchez, Iñigo Tamayo, Igor G. Olaizola
- **Journal:** Multimedia Tools and Applications
- **Publisher:** Springer
- **Year:** 2014
- **DOI:** http://dx.doi.org/10.1007/s11042-013-1516-7

**Abstract** - Digital home application market shifts just about every month. This means risk for developers struggling to adapt their applications to several platforms and marketplaces while changing how people experience and use their TVs, smartphones and tablets. New ubiquitous and context-aware experiences through interactive 3D applications on these devices engage users to interact with virtual applications with complex 3D scenes. Interactive 3D applications are boosted by emerging standards such as HTML5 and WebGL removing limitations, and transforming the Web into a real application framework to tackle interoperability over the heterogeneous digital home platforms. Developers can apply their knowledge of web-based solutions to design digital home applications, removing learning curve barriers related to platform-specific APIs. However, constraints to render complex 3D environments are still present especially in home media devices. This paper provides a state-of-the-art survey of current capabilities and limitations of the digital home devices and describes a latency-driven system design based on hybrid remote and local rendering architecture, enhancing the

interactive experience of 3D graphics on these thin devices. It supports interactive navigation of high complexity 3D scenes while provides an interoperable solution that can be deployed over the wide digital home device landscape.

**Keywords** - Home device interoperability, Digital home applications, Computer graphics, 3D virtual environments, Interactivity, Hybrid rendering system

### 3.2.1 Introduction

Users are becoming more accustomed to improved experiences that provide interactive 3D applications exploiting the technology in immersive environments. Thanks to the advent of low energy-consumption Graphic Processing Units, interactive 3D applications are currently running in most digital home devices. Connected TVs, smartphones and tablets are being fitted with graphic capabilities providing users an enhanced experience on top of interactive and 3D applications, and pushing the market to new advanced 3D applications with complex interactive virtual environments.

The landscape of digital home devices has changed last years completely with the introduction of smartphones and tablets in the home network bringing secondary displays to foster customized media, together with the evolution of the TV to Smart Connected TV. Moreover, these kind of devices are running over application-based Operating Systems. Most popular are Android and iOS [TL10] for smartphones and tablets and different proprietary platforms (Samsung, Philips, etc.) for Smart TVs. However big companies such as Google or Apple offer a Connected TV solution which could nearly provide a full digital home approach through the different devices and their Operative System. Each solution facilitates a framework and an SDK (Software Developer Kit) to exploit native assets providing the hardware features of the devices: connectivity, motion and voice control, camera, GPS, graphic capabilities, etc. However, the deployment of the applications from one OS to the others implies major changes and specific adaptation. This platform heterogeneity at the OS level generates an important interoperability problem.

The rapidly increasing use of the Web as a software platform with truly interactive applications is boosted by emerging standards such as HTML5[1] and WebGL[2] that are removing limitations, and transforming the Web into a real application platform middleware to tackle the interoperability problem. Following this

[1] Html5 standard specification (May 2011) http://www.w3.org/TR/html5/
[2] Webgl website (Mar. 2011) http://www.khronos.org/webgl/

trend, the new HbbTV[3] standard for broadcasting environment interactivity is also based on a specific HTML browser.

HTML5 provides devices the capability to run rich web applications accessing the entire device features on a web browser. It comes together with CSS[4] and JavaScript which provides an appropriate framework for the content interactivity and universal access to different APIs. WebGL is the API oriented to 3D graphics in the HTML5 canvas element. It is easier to craft innovative user experiences using powerful HTML5 layout and WebGL rendering engines than current native IDEs.

Digital home browsers are rapidly adopting HTML5 features on a tough race just after the desktop browsers. The standard has won a prominent place as a horizontal approach to reach interactive multimedia applications on home devices. HTML5 applications can be packed for the different execution environments providing an interoperable application with minor changes through different OSs. That is why HTML5 is being strongly promoted by the standardization bodies and a sector of the market to achieve a HTML5 marketplace instead of the different proprietary ones, such as Android Market, iPhone App Store, Samsung Apps Market, Net TV Apps, etc.

Digital home applications are changing how people experience and use these devices. The incoming pioneer interactive 3D applications for mobiles are inciting users to discover new ubiquitous and context-aware experiences through smartphones and tablets and show the feasibility to access this rich media apps through the Smart TV. User requirements are involved in the mentioned tough race demanding power efficient techniques together with advanced interactive virtual applications with complex 3D scenes on digital home devices as they do on PCs.

The introduction of the canvas element into HTML5 enables 3D rendering on the Web while WebGL technology brings hardware-accelerated 3D graphics to the Web Browser without plug-ins turning HTML5 into the promising solution to cope with such fragmented device market by universal developments for device-independent applications and services. This paper provides a complete state-of-the-art of the current browser capabilities of the digital home devices

[3] HbbTV 1.5 specification (April 2012) http://www.hbbtv.org

[4] Cascading style sheets (css) standard specification (May 2011) http://www.w3.org/TR/CSS/

using HTML5. We present performance results concluded by experiments carried out in representative set-top boxes, smartphones and tablets. The current limitations to run advanced interactive 3D applications are also explained in the article giving rise to a system proposal to overcome the detected handicaps to be able to run advanced interactive 3D applications using HTML5, making thin devices suitable for a wider range of applications. A system architecture called *3DMaaS* is detailed to provide complementary rendering capabilities to these devices, adding to their own capabilities the chance to push to the cloud complex 3D rendering tasks. A technical validation of *3DMaaS* is done emphasizing on the overcoming of the limitations detailed on the state-of-the-art.

### 3.2.2 Digital Home Device Software Platforms

The TV is still the main device for watching media content in the digital home. Nevertheless in the same way that mobile phones have gone from thin to smartphones and tablets, providing access to all kind of services and contents, home television is evolving from a passive device for multimedia content consumption to the so called SmartTV. Worldwide shipments of Internet-connected televisions have reached 25% of total units in 2011 and it is expected that it will the 70% by 2016[5].

However, the Connected TV platforms are very heterogeneous and based on proprietary approaches, where the interoperability is a problem. TV manufacturers have developed their own frameworks, providing a SDK to develop specific applications and with a own marketplace. Samsung Smart TV provides a SDK to develop Flash-based or JavaScript engine-based applications. These applications are located by Samsung in their marketplace called Samsung Apps. Philips Net TV provides a CE-HTML browser with a index page to access to the Net TV apps. Connected set-top boxes are also very heterogeneous with different web browser such as Opera Mobile, specific OS such as Boxee[6] or burgeon Linux/Android devices[7] to transform not-connected TVs into a full connected devices. Moreover, Google and Apple have their TV solutions, Google TV[8] and Apple TV[9] respectively, but they are not positioned yet as a market leader as they do on mobile systems.

[5] May 2012. IMS Research.
[6] http://www.boxee.tv/
[7] http://www.raspberrypi.org/
[8] http://www.google.com/tv/
[9] http://www.apple.com/uk/appletv/

Smartphone and tablet market penetration is going faster than Connected TVs. In Q3 2012, global smartphone shipments jumped to 179 million units[10]. It was a rise of 45% from last year beating the annual growth rate. Growth continues but it is slowing down as most of the developed markets come close to 80-90% penetration. Meanwhile, global tablet shipments reached 20 million units in Q3 2012[11]. As tablets and smartphones get faster, allowing a quicker transfer of data, integrating new connectivity and interactivity paradigms along with fancy graphics, users have developed a habit for downloading applications. This pushes mobile application market to a rapid evolution shifting the business landscape and to a competitive environment. The research firm Gartner recently forecast that mobile application stores will deliver 310 billion downloads internationally in 2016 and $74 billion in revenue[12]. A key difference of each platform is the application market, App Store, Android Market and Windows Phone Marketplace. Hereby, Gartner claims that an integrated cross-device experience will help fuel this demand.

The Android Market is open [But11], whereas others are gated. This means, Android foster developers to self-publish created applications into the Android Market, whereas Apple or Microsoft decide what gets published keeping the application approval right before they become available in the Marketplace.

The different marketplaces availability responds to the change on the mobile phone landscape, playing their correspondent OS, such as iOS, Android, Windows Mobile or Research In Motion (RIM) BlackBerry OS, a prominent role in the applications development [TL10]. The market penetration of Android and iOS is increasing strongly and both are becoming the two major OSs to take into account. While on January 2011 the market share of Android and iOS was 61% in Europe, on December 2012 it has raised up to 85% thanks to the big increase of the Android OS. This trend is more representative in North America, where on January 2011 the Android and iOS mobiles represented the 61% of the market, and two years later they are over the 90% (Figure 3.1).

The Android operating system is built from a modified Linux kernel. Previous specific versions for tablets and smartphones, entirely designed for devices with large screens and thinner devices respectively, converge in the version 4 that brings together phones and tablets easing the multi-device development

[10] Kang, T.: Global smartphone vendor market share: Q3 2012. International Data Corporation (Oct. 2012)

[11] Mawston, N.: Global tablet vendor market share: Q3 2012. Strategy Analytics (Oct. 2012)

[12] Market Trends: Mobile App Stores, Worldwide. Gartner Sept.(2012)

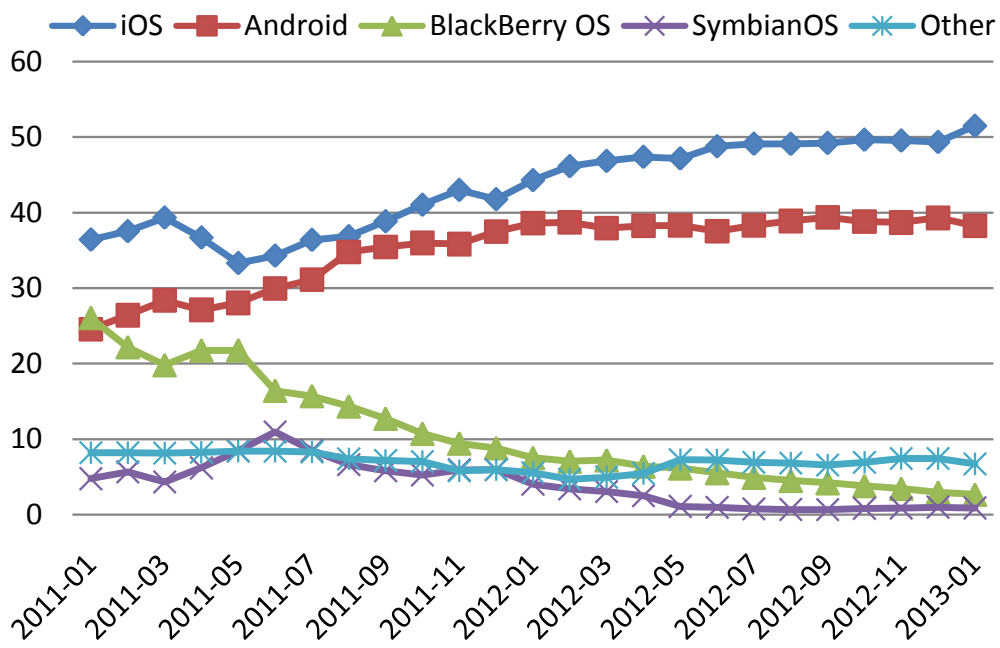


**Figure 3.1:** Top Mobile OSs in North America from Jan 2011 to Apr 2012 (StatCounter Global Stats)

and interoperability. The software stack contains Java applications running on a virtual machine, and system components are written in Java, C, C++, and XML. In order to develop Android applications the SDK can be integrated in different environments such as Eclipse.

Apple developed the iOS for its products catalogue. The operating system is derived from Mac OS X and is built on top of the Darwin foundation and XNU kernel. XNU combines the Mach 3 microkernel, elements of Berkeley Software Distribution (BSD) Unix, and an object-oriented device driver API (I/O kit). iOS frameworks are written in Objective-C. In order to tackle application development for iOS, Apple provides a Xcode development environment and a iOS Simulator to test applications.

To sum up, the current digital home platform ecosystem is heterogeneous, with several operating systems, programming languages, and interfaces, resulting in more complex software cross-platform development and testing processes. Digital home devices increasingly depend on reliable software to offer a fresh user experience. Hence the current trend in developing interoperable applications lays on using Web technology instead of platform-specific APIs.

### 3.2.3 The Web as a Software Platform

According to the wide landscape of digital home application frameworks described in the previous section, developers need to carefully determine how and where to invest their time and effort before tackling an application development project. Writing native applications requires developers expertise and background in specialized IDEs.

However, Smart TV, smartphone and tablet trend is to be always connected to the Internet. Application developers should not ignore advantages of moving from desktop computing to web-based applications [TMAS11] [TM11] [ASMT11]. On the one hand, applications provided on the Web as services do not require installation or manual upgrades, easing the software life cycle management while inherit web security and privacy policies. On the other hand, in terms of monetizing an application, another relevant advantage lays on the deployment and sharing of Web applications that can be instantly worldwide, with no middlemen or distributors. This way the application monetizing strategy do not have to obey marketplace policies enabling a free design of the business model. Last but not least, the potential of the web-based applications can support user collaboration over the Internet, deploying virtual spaces where users interact and share application experience and data, fostering new paradigms of interactivity and social networking.

Developers can also benefit from web-based solutions saving time and effort. They can apply their knowledge of designing web applications to smartphone, tablet or TV application design, removing learning curve barriers. Web-centric approach for digital home applications enables not only rapid prototyping, but also unified integration with Web services. It requires access to the hardware resources of the digital home devices through JavaScript that always lags behind the new capabilities that manufacturers introduce. In order to mitigate this limitation new W3C's HTML standard provides device orientation, speech recognition and geolocation management bridging from native features to web-centric development.

From the viewpoint of the developers, the key for transition towards web-based software is the ongoing evolution of web development technologies, specifically HTML, CSS and JavaScript. This way, development turns more efficient to face interoperable and innovative mobile user experiences exploiting powerful HTML layout and rendering engines than native IDEs.

Emerging standards such as HTML5 and WebGL will play a crucial role removing the remaining limitations and transforming the Web into a horizontal software platform. They will significantly shift the perception of the web browser and web applications capabilities to a fully featured web-centric operating system and to a fully interoperable application respectively.

W3C HTML5 standard specification[13] defines the core language of the World Wide Web. New features and elements are introduced paying an special attention to improve interoperability.

HTML5 provides many capabilities enabling developers to combine video, audio, 3D, and 2D into one seamless application. HTML5 embraces multimedia by means of built-in audio and video support through <*audio*> and <*video*> tags that allow media files to be played without third party browser plug-in components. Moreover, specifically for live multimedia streaming Dynamic Adaptive Streaming over HTTP formats rise as the solution to provide high quality video streaming on the Internet thanks to enabled adaptivity. To sum up, adaptive HTTP streaming is a promising technology to overcome access to media consumption through home network devices facing the bitrate and resolution adaptation to each singular context while manage seamless underlying network topology. Thereby, Google Chrome, Opera, Safari or Firefox browsers bet on Adaptive HTTP Streaming formats including them on their development roadmaps. There are different proprietary implementations such as Microsoft Smooth Streaming, Apple HTTP Live Streaming (HLS) or Adobe HTTP Dynamic Streaming. But MPEG-DASH had been accepted by ISO as an International Standard with the purpose to converge all the proprietary approaches into the standard.

HTML5 also brings relevant features fostering new paradigms of interactivity and user experience. The Canvas API provides salient 2D drawing capabilities for interactive graphics. Moreover, HTML5 specification provides numerous additions and enhancements such us realtime message based, speech recognitions, device orientation awareness or drag&drop action to be applied to the HTML objects.

CSS3[14] brings lots of possibilities that boost creativity such as transitions, opacity definition and native columns. It also provides much more flexibility enabling 3D effects such as zoom, pan, rotation, transformations and animations.

[13] Html5 standard specification (May 2011) http://www.w3.org/TR/html5/

[14] Cascading style sheets (css) standard specification (May 2011) http://www.w3.org/TR/CSS/

But the most relevant features that turn HTML5 into a interoperable software application platform are:

- Offline operation. The HTML5 contains several features that address the challenge of building web applications that allow to operate even when an active network connection is not available.

- Local storage. HTML5 brings a persistent cache based on local SQL database, allowing data to be stored locally in the device. It also provides a filesystem API in order to manage read and write actions.

Moreover, HTML5 development can be easily transformed in an application package ready to be provided in the Android marketplace or in other App stores. PhoneGap[15] is an open source framework for creating mobile web applications in HTML5, JavaScript and CSS3 while still taking advantage of the core features of native applications in some platforms such as iOS and Android devices.

Applications often engage users through 3D visual interfaces. They are more effective, attractive, and are considered as a key factor to add value to applications improving the overall user experience. For the Web, WebGL takes the role of enabling technology as a solid foundation for 3D graphics applications [OJ10].

WebGL[16] is a cross-platform web standard for hardware accelerated 3D graphics API developed by the Khronos Group that includes among others Mozilla, Apple, Google and Opera. WebGL brings to the Web the support to display and manipulate 3D graphics natively in the web browser without any plug-in components. WebGL performs 3D graphics on top of the HTML5 canvas element and is accessed using Document Object Model (DOM) interface. WebGL allows communication between JavaScript applications and the OpenGL software libraries, which accesses the graphics processor of the device. This makes possible to exploit hardware capabilities to render 3D content. WebGL is based on OpenGL ES 2.0, and it uses the OpenGL shading language GLSL.

### 3.2.4 HTML5 Interactive 3D Applications

#### 3.2.4.1 Advanced 3D Application requirements

Recently the use of 3D graphics in many industrial fields and applications such as games, advertisement products interaction, serious gaming/simulation for more

[15] Phonegap website (Jan. 2012) http://www.phonegap.com
[16] Webgl website (Mar. 2011) http://www.khronos.org/webgl/

effective training, financial and medical data analysis, and CAD design are increasing more and more. Often applied data 3D applications interfaces exploit 3D graphics to support professional user productivity or represent data that could not be done otherwise such as Google Earth. But 3D graphics is also used for making more visually attractive interfaces.

Due to the mobility of users and professionals involved in these applications, it is mandatory to provide access through mobile devices tracking the variety of contexts of the user. Facing the emerging trend in consumer technology for delivering 3D content to the mainstream user via digital home devices[Ort11], new solutions must promote the communication between the TV and mobile devices of the digital home. This provides users access to 3D graphics applications through Connected TVs, smartphones and tablets having a great experience interacting with 3D virtual environments. Demand for 3D visualization is increasing in these devices as users expect more realistic immersive experiences. So 3D graphics combines immersion and interactivity fostering creativity for new envisaged applications and information navigation interfaces.

Mobile games are one of the fastest growing segments of the application industry. Bringing together the social gaming paradigm and the internet connection capability of most of the digital home devices, users will embrace the online interaction trend from PC.

User interfaces based on 3D graphics let users interact with virtual objects, environments, or information but the experience can be improved with the inclusion of real media sources around the user. According to [Azu97] definition, virtual worlds technologies completely immerse a user inside a synthetic environment. In contrast, augmented reality allows the user to enjoy the real environment, with virtual objects superimposed upon or composited with the real world providing extra information or interactivity about what is around. This requires the fusion of very heterogeneous media sources in a concept called 3D Media [DA09] [ZDLB08]. 3D Media is composed of different audio and video sources, static images and 3D objects enabling enhanced experiences.

3D rendering pushes the visual boundaries and interactive experience of rich environments, but 3D interfaces, virtual worlds and augmented reality applications require high 3D graphical features. The more complex the 3D scenes are, the higher the hardware requirements are. Although connected TVs, set-top boxes, smartphones and tablets are rapidly improving their graphic capabilities thanks

to the integration of low energy-consumption Graphic Processing Units, the capabilities are below the user expectation. Users are demanding experiences they are used to in powerful devices such us PCs, mixed with the new characteristics that the digital home devices provide, such us using the camera of the smartphone for an ubiquitous augmented reality experience.

#### 3.2.4.2 Limitations of the Browsers in Digital Home Devices

WebGL API coupled with JavaScript engines are boosting increasing capabilities of the web browsers making possible to develop complex computational environments including 3D graphics. Therefore platform-independent applications are directly performed through the web browser on different devices without the need to install additional software or plug-ins bringing the accessibility and interoperability of the web. However, constraints to render complex 3D environments are still present in digital home devices. It is necessary to define the hurdles, in terms of performance, that a developer will face when creating a web browser-based software for 3D interactive applications on top of HTML5 and WebGL. Here, we introduce not only the limitations around complex applications that require 3D graphics technology according to a set of evaluations, but also the clues about the bottlenecks origin that would enable the work to be done to remove the detected barriers.

We have chosen different devices that provides a wide representative landscape of the current digital home platforms, in order to detect the browser capabilities and limitations. On one hand, we have selected two high performance set-top boxes with browsers that support HTML5 and WebGL:

- **Innout Media Center 4Gs HD Set-top Box**: Opera Mobile 12.0 browser supporting HTML5 / WebGL profile and HbbTV profile.
- **Gigabyte GN-SB100 series**: Android 2.2 OS, Opera Mobile 12.0 browser.

These set-top boxes support WebGL but do not have specific hardware to run it, so they can not deal with it at all. However, Opera Mobile has announced[17] that its TV browser with WebGL runs on the recently launched Intel Atom Media Processor CE5300. Mitsubishi Electric is also working on a set-top box with a high performance TV browser called Espial[18] with WebGL applications support.

[17] March 2012. IP&TV World Forum in London.

[18] March 2012. http://www.espial.com/company/press_item/id\discretionary{-}{}{}745

On the other hand, according to the mobile devices in the digital home, two of the selected devices are Android and the other two are iOS. In order to track the market trend, where the tablets have an increasing presence, the evaluations consider two smartphones and two tablets:

- **Samsung Galaxy S**: GT-I900 smartphone with Android 2.2.1 firmware.
- **Samsung Galaxy TAB**: GP-P1000 tablet with Android 2.2 firmware.
- **iPhone 4**: iOS5 smartphone.
- **iPad**: iOS5 tablet.

Android and iOS Safari default browsers do not support WebGL yet. Neither Opera Mini nor Google Chrome Beta version for Android 4 do, but all of them have included it in their roadmaps. Here, for the Android devices, the Mozilla Firefox 4.0 browser have been employed for the tests. Firefox has WebGL support and it can be installed from the Android Market. Other browsers such as Opera Mobile 12.0 also support WebGL for Android devices. But for the iOS devices, a specific application which runs a webkit based browser called GoWebGL[19] provides WebGL capabilities.

In order to measure the frame rate achieved for each device, a simple 3D scene is composed using rotating cubes. In each test the total number of 3D objects is increased as well as their polygonal complexity, ranging from 1 to 80 objects and from 12 to 200k polygons per object. The geometry of one single cube is loaded into a vertex buffer which is drawn multiple times using a different transform matrix for each cube. The performance is measured as the average time per frame sampled over 50 frames for each object and polygon configuration. Figure 3.2 shows the results of these tests as the maximum number of polygons that can be rendered in interactive time (15 fps) in function of the object quantity in the iPad.

As seen in the plot, the maximum number of polygons drops exponentially with the number of 3D objects. Given that the geometry is loaded as a vertex buffer and provided that the total number of polygons is maintained constant, these results can be explained by two reasons. On the one hand, floating point operations are very CPU demanding in JavaScript. Since an additional cube means an additional matrix rotation, the overall performance is significantly affected. On the other hand, the new transform matrix must be transferred to the GPU

[19] https://github.com/gauthiier/GoWebGL

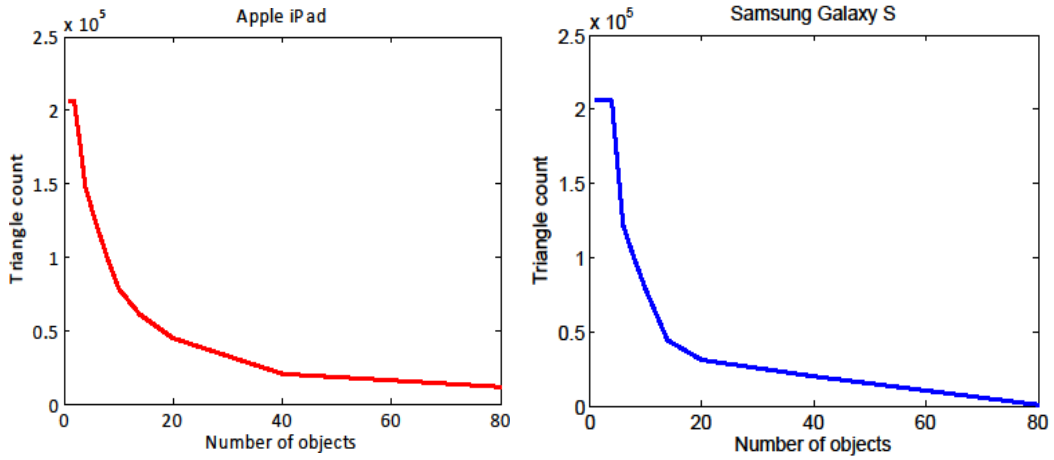


**Figure 3.2:** Maximum number of polygons that can be rendered in interactive time (15 fps) in function of the object quantity in the iPad and Samsung Galaxy S

overloading the CPU-GPU communication bus. Although the amount of data is quite small, the bus latencies of these small devices can have a negative impact which means a notable bottleneck.

From these results it can be derived that current WebGL subsystems can support a good performance for simple scenes composed by small amounts of objects, regardless of its polygonal complexity. This limitation brings an important drawback hindering scene-graph based rendering engines, since each object in the graph must be transformed recursively with respect to its parent.

Figure 3.3 compares the performance of the different devices in terms of the maximum number of polygons that can be rendered against the number of objects while keeping 15 fps target frame rate. The trend is quite similar through the four devices. The iPad and iPhone have the same 3D processing behavior while increasing the number of objects, being the performance of the iPad slightly better than others. The Android devices achieve almost the same throughput while increasing the number of objects and the responsiveness is very close to the iPad. However, the capabilities drop from 40 objects, specially in the Samsung Galaxy S.

These results become evident the need to improve the performance of 3D applications over the digital home browsers. However, these measures where unachievable some months ago, and the rapid adoption of HTML5 features on the mobile browsers let us think that these results are going to be improved very fast removing barriers in terms of WebGL compliance. Android and iOS browser will be able in the near future to run WebGL in the same way that other mobile

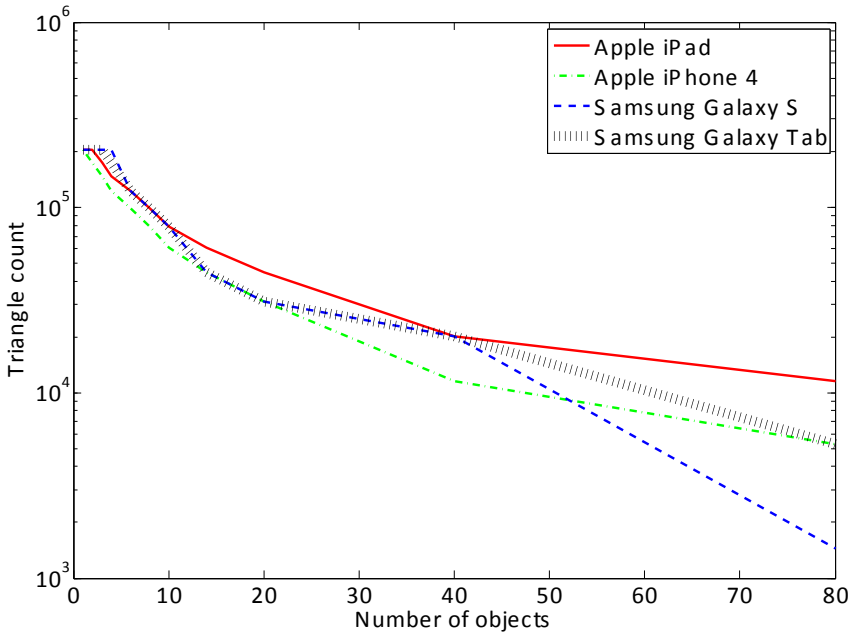


**Figure 3.3:** A comparative of the maximum number of polygons that can be rendered in interactive time (15 fps) in function of the object quantity in the different devices

browsers will do it (Google Chrome, Opera Mini, etc.) and will be accessible from these platforms. The WebGL performance itself needs to be improved by a better integration of the JavaScript capabilities of the browser and the architecture of the device. However, remaining throughput limits closely related to GPU potential would not disappear quickly due to life battery technological constraints.

In terms of HTML5 and WebGL support for Connected TVs and set-top boxes, different initiatives such as Espial or the Opera Mobile for TV highlight the relevance of these technologies on the roadmaps of the TV browser developers.

Anyway, according to the obtained results, and even if the capabilities of the devices are going to increase rapidly, users are already demanding advanced 3D applications on digital home devices. The proposed *3DMaaS System* faces all the previously described issues responding those who are not willing to upgrade their devices as fast as the market moves. It also optimizes the development investment of a new application turning it suitable for any device with video streaming support. Moreover, these results establish thresholds to define 3D performance profiles to support local/remote rendering distribution decisions according to the 3D scene complexity. This way *3DMaaS System* can mitigate local 3D processing stress of the device by taking care of full or partial 3D rendering in a remote resource that is real-time encoded and streamed inside a video.

Section 3.2.5 shows the related work on different approaches to increase the capabilities of the devices to render 3D content and introduces the *3DMaaS System* proposed in section 3.2.6, which allows to extend the capabilities of the devices pushing to the cloud complex 3D rendering tasks and combining it with its hardware possibilities on a hybrid system.

### 3.2.5 Related Work

A solution based on remote rendering performed by a high processing cloud server with enough network bandwidth resources can keep the target performance while achieve interoperability widen the audience. The server would manage all the 3D Media involved in order to render the 2D result according to the user actions. Last but not least, standard mechanisms to adapt the video stream to the network capacity can solve bandwidth problems. However, this solution delegates the final performance to interaction latency. Different approaches driven by the described solution, face digital home device's applications to overcome the current limitations in terms of 3D processing and rendering.

The gaming sector is the main driver for graphics computation. Recently launched cloud hardware solutions, such as by Nvidia Grid product[20], brings promising cloud-based video streaming technology ready to deliver gaming content to consumer devices, including smartphones, tablets, PCs, and TVs enabling up to 36 concurrent HD-quality video streams with low latency from a single server using NVIDIAs GPU virtualization technology. Nvidia Grid faces GaaS boosting such as OnLive[21] or Gaikai[22], lately incorporated to Sony, to overcome scalability and performance issues.

The concept of Gaming as a Service (GaaS) is presented on [MTP12] where the quality of experience and the latency are key factors of success. These features can be dramatically enhanced when combining the computational load of the local machine with remote rendering by sending complex calculations to a remote server using proprietary approaches. [LLJ+09] proposes the Games@Large System oriented to set-top boxes on home networks and for enterprises such as hotels. [FE10] extends the Games@Large System with the main idea to calculate motion vectors directly from the 3D scene information used during rendering of the scene.

[20] http://www.nvidia.com/object/cloud-gaming.html
[21] http://www.onlive.com/
[22] http://www.gaikai.com/

Similar hybrid computation approaches also tackle visualizing 3D objects on other sectors. These solutions consist on sending graphical commands such us roto-translation parameters from the end client to the server. This way the server can calculate the strictly necessary data that the end client needs and stream it offering a progressive reconstruction of the polygons. These solutions are valid for a mere combination of 3D objects, but not extensible for 3D Media based applications.

[LS07] proposes a remote rendering scenario for mobile devices like PDAs running a dedicated application called Mobile 3D Viewer. This approach is based on the Chromium software [HHN+02]. [MGV+10] presents an approach sending 3D graphical commands in a stream from the server to the client and it is based on WireGL [HEB+01].

SHARC System [SLLE10] is an approach for enabling scalable support of real-time 3D applications in a cloud computing environment. It is based on service virtualization with tools like VNC. This solution extends VNC as a video streaming platform. VNC and similar virtualization tools are also used on [SDTDD09] and [TKBS14].

[NCB06] presents a MobiX3D mobile player for access 3D content through mobile devices using OpenGL ES.

Contrary to the above described solutions our approach does not require a specific player o application on the client side, running on a HTML5 browser to overcome interoperability. In section 3.2.6 we present the *3DMaaS System* which exploits the potential of WebGL, based on OpenGL ES 2.0, leveraging 3D processing on mobile devices by delegating 3D WebGL rendering to a remote server. *3DMaaS System* enables the 3D Media content based applications by means of adaptative video streaming from the server side to the end device.

### 3.2.6 3DMaaS System Design and Experiments

#### 3.2.6.1 Design

There are three main actors on the *3DMaaS System* [ZdPCU+11] (Figure 3.4): MaaS Manager (MM) which monitors the computational load of the resources pool and dispatchs the device request to one of them to achieve a target QoS through load balancing strategies; Rendering Server (RS) the remote rendering resource; and their communication with the end devices. The features that *3DMaaS System* requires are really affordable for any kind of end device. Moreover, the

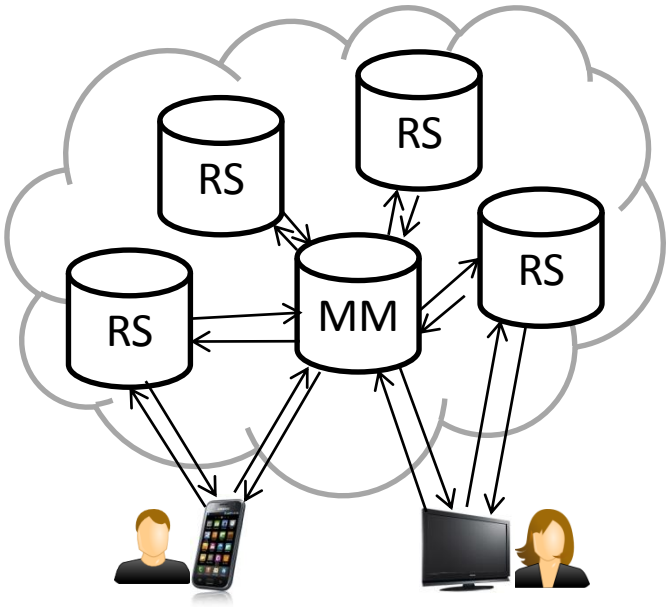


**Figure 3.4:** General infrastructure of the *3DMaaS System*

cloud rendered stream is adapted to the different codecs and parameters to represent the media content at the different end devices (set-top boxes, smartphones, tablets, etc.). A block diagram of the general architecture is shown in Figure 3.5 and all the modules are more deeply explained below.

*RS* is the core module of the *3DMaaS System*. Figure 3.5 shows the different blocks of the RS and its communication with MM and the end device:

- **Web services with MM**: MM reports end device context to the RS.
- **Internal manager**: It manages the requests and creates the streams.
- **3D Media & Render**: According to the real-time captured context such as object user interaction it generates the rendering for the composition.
- **Streaming server**: It deals with real-time encoding and the streaming session with end device taking into account the context profile captured/negotiated by the MM: Device features (supported streaming protocols and codecs, screen size, etc.); Connection context (network bandwidth, etc.); and User preferences (objects in the composition, their size, etc.).
- **Web sockets for user real-time interaction**: TCP web sockets are used for real-time communication. The content user interaction is translated to: changes on the composition (add new elements, delete them, move their

position, resize them, etc.); modifications over an object (3D movements, texture changes, stop or rewind a video or audio, etc.); and adjustments of the streaming parameters (video resolution, bitrate, codec, etc.).

The *3DMaaS* System aims a wide range of video streaming formats in order to fit in very different devices. To achieve it *3DMaaS* provides a complete set of streaming formats [ZMM+12], dealing with RTSP and Dynamic Adaptive Streaming over HTTP [HFF11] such as HLS and MPEG-DASH. The *3DMaaS* Streaming Server must launch a suitable pipeline according to the previously negotiated format because each alternative is supported depending on the browser implementation[23]. Open Source frameworks provides the pillars to the 3DMaaS Streaming Server. Being more specific, Gstreamer performs RTSP server and some plugins[24] [25] bridge HLS communication, while GPAC[26] and DASH-JS[RLMT12] JavaScript- and WebM-based DASH library for Google Chrome hold MPEG-DASH compliance. Last but not least, x264 tune options[27] accomplish the required ultra low latency that keeps a good interaction latency to guarantee the quality of experience of the user.

In terms of achieving low latency, the main solutions deployed lays on: Web socket for application logic communication and system awareness of user interaction; video codec tuning to push the streaming processing time to the minimum; multimedia encapsulator set up to minimize the buffering requirements; RTCP session, for those suitable streaming protocols, in order to perform quality of connection measures enabling dynamic streaming parameter settings to keep QoS.

Concerning scalability, *3DMaaS System* size is a critical factor because it must provide enough RS resources to satisfy the incoming demand of remote rendering service. To face it, *3DMaaS System* has been designed to ease the rapid deployment of new RS instances but an automatic elastic behavior according to usage forecasts is out of focus.

Regarding the end device, the capabilities required by *3DMaaS System* for the HTML5 application of the client are really affordable for most of the common digital home devices. It only has to include *video* tag with the video streaming address provided by the RS and scripting capabilities to send HTTP interaction

[23] http://www.longtailvideo.com/html5/
[24] http://gitorious.org/ylatuya-gstreamer/gst-plugins-bad/commits/hlswip
[25] https://github.com/ylatuya
[26] http://gpac.wp.mines-telecom.fr/2012/02/01/dash-support/
[27] http://mewiki.project357.com/wiki/X264_Settings

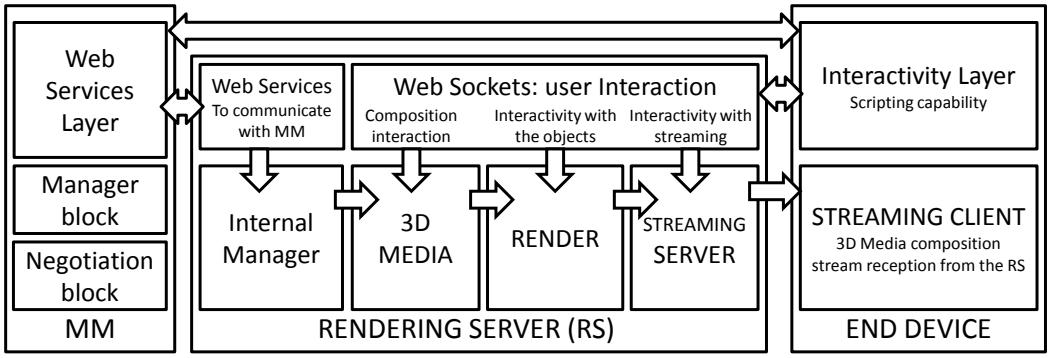


**Figure 3.5:** The block diagram of MM and RS and their communication with the end device

parameters. Their target is twofold: establish a new connection with *3DMaaS System* on a initial negotiation through MM; and for delivery of TCP web socket requests for low-latency interaction once the streaming communication is running with the RS.

#### 3.2.6.2 **Experiments**

The critical performance metric of remote rendering solutions is the experienced latency for delivering a frame after graphics rendering update driven by user interaction. Our approach based on a video streaming server for 3D interactive application overcomes latency challenge. It is tackled by the hybrid solution proposal combining remote rendering of background 3D objects, where latency does not have a high impact on the user experience, with local browser WebGL rendering of foreground 3D objects which require low latency. The application server processes the user input and renders new screen frames and transmits them to the device in real time. Moreover, [ZMM+12] presents *3DMaaS System* results for low latency streaming applications achieving 27.84 ms latency score. The hybrid strategy minimizes the number of objects that the browser have to render optimizing performance. For this, various experiments were carried out in order to assess the efficiency of the proposed architecture for visualization of 3D scenarios through digital home browsers. Users interact with the 3D rendering applications running on an accelerated graphics back-end for remote rendering and web browser for local rendering, allowing highly interactive experiences regardless of the complexity of the scene being considered.

Here, a low quality connection of the end device would have a negative impact on latency. To mitigate it and keep the Quality of Experience, the streaming

session is monitored and dynamically modified in terms of bitrate and framerate. Since visualization framerate experienced at the mobile client constitutes the main limitation of 3D web based applications, especially when considering complex 3D scenes, framerate driven analysis tests have been designed in order to accurately quantify critical parameters of our hybrid visualization system, thus providing an effective measure of the performance of the proposed architecture.

Unfortunately, none of current available TV sets are not able to deal with 3D rendering tasks. These devices cannot perform 3D WebGL applications due to lack of specific hardware but can also benefit from the *3DMaaS System* pushing to the cloud the whole rendering scene instead of building an hybrid rendering approach. Therefore, the experiments performed to define the performance thresholds on hybrid scenarios have been focused on mobile devices.

The tests have been done over the same devices described in section 3.2.4.2 in order to measure the frame rate achieved for each device with the *3DMaaS System*. But in this case two superimposed HTML5 canvas have been involved. The one on the front is the simple 3D scene described on section 3.2.4.2, composed by rotating cubes. The canvas in the back is a <*video*> tag receiving a live video stream from the remote rendering server with the 3D background.

In order to set up the tests, the same range that defined in section 3.2.4.2 has been employed for the number of 3D objects as well as their polygonal complexity in the front canvas. This way the performance combining the 3D local rendering capabilities and video stream reception on the different mobile devices is compared with the obtained measures on section 3.2.4.2 with a mere local 3D rendering.

Figure 3.6 compares the frame rendering time for a number of polygons performed in a Samsung Galaxy TAB including the 3D object canvas and the live video stream visualization, with the measures obtained for the same 3D scene without the background video stream. The aggregation of the remote rendered live video stream does not have a considerable impact on the performance adding just an extra constant CPU demand. This way, rendering time for advanced applications with demanding 3D capabilities are not penalized by the added video stream. The GPU turns into a bottleneck from $10^6$ triangles for this simple 3D scene, so this barrier settle the complexity that can be afforded by the device GPU without performance drawbacks. From this point remote rendering would make possible complex scenarios with no GPU overhead keeping the interactivity

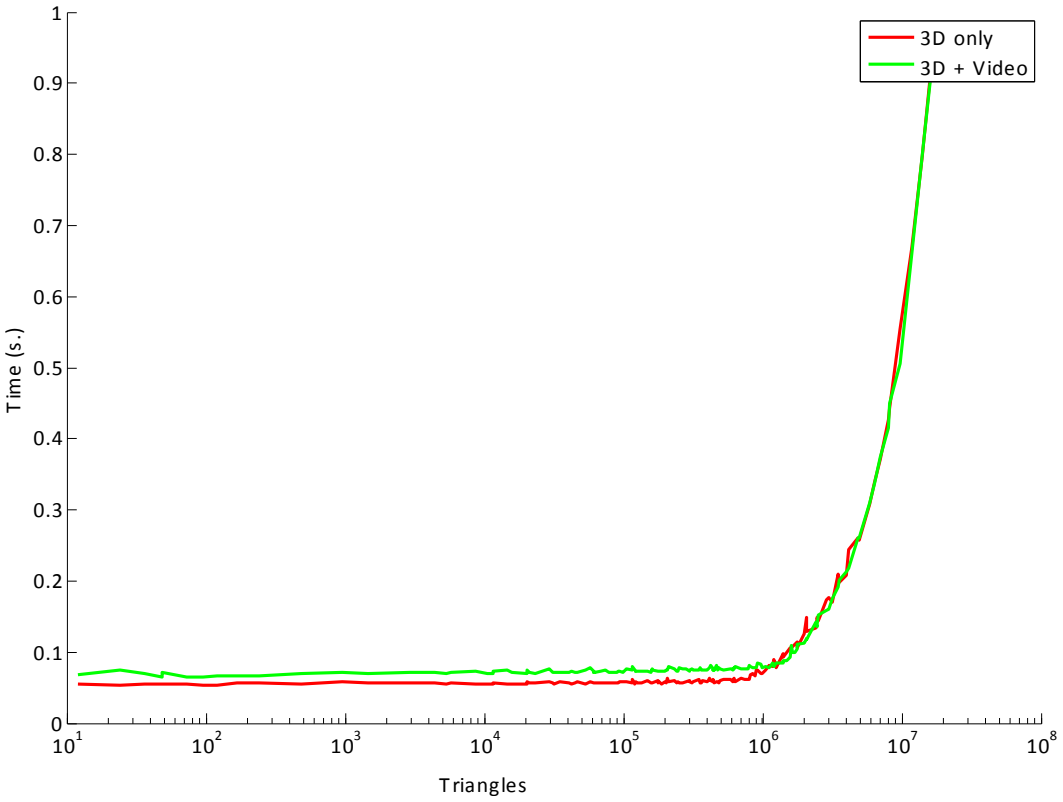


**Figure 3.6:** A comparative of the frame rendering time and the number of polygons rendered in the Samsung Galaxy TAB, for 3D contents and added remote rendered live video stream to the 3D contents

performance of the application. So this approach provides the application enriched 3D rendering capabilities, extending the device's hardware through remote rendering.

The results obtained by the proposed architecture for hybrid remote and local rendering enhance the interactive experience of 3D graphics on digital home devices, proving the feasibility of interactive navigation of high complexity 3D scenes while provides an interoperable solution that can be deployed over the wide device landscape. However, this approach transfers responsibilities related to synchronization and OpenGL state consistency of local and remote 3D scenes to the application.

Figure 3.7 depicts different services and games from pure 3D object interaction in games, for e-learning purposes, to on demand content delivery services, specifically driven to e-inclusion and entertainment, which have been deployed on top of the *3DMaaS* infrastructure comprising the system portfolio.

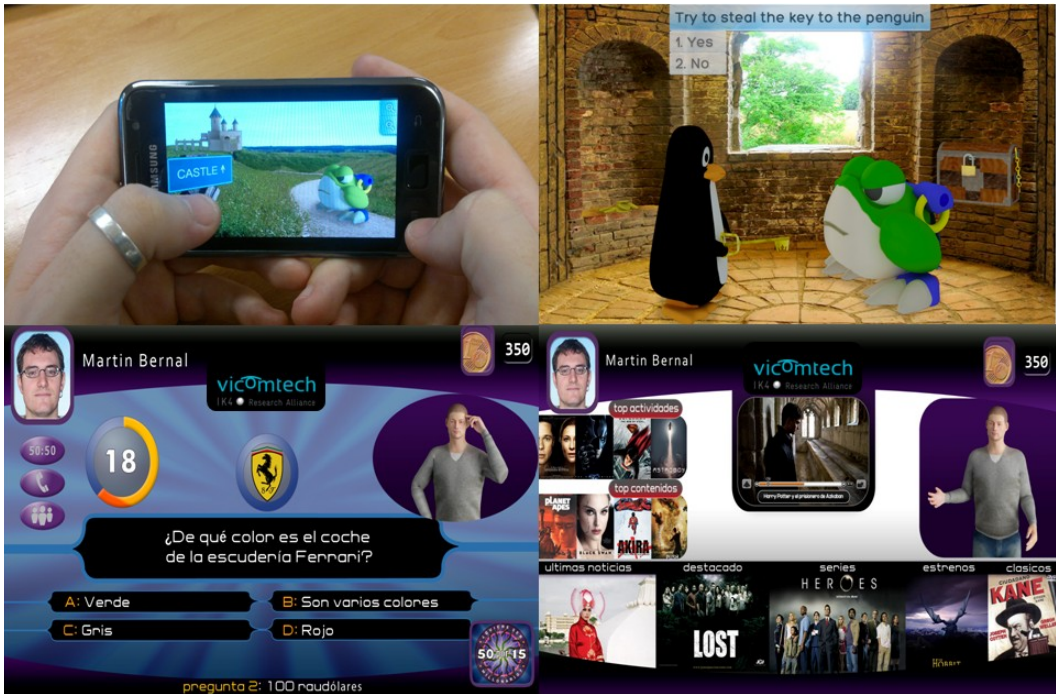


**Figure 3.7:** Set of 3D Media contents delivered through 3DMaaS

### 3.2.7 Conclusions

Digital home application is a very disruptive market overcoming the potential that the Internet has and the incorporation of mobile devices together with the evolution of the TV to Smart TV in the digital home. These devices increasingly depend on reliable software to offer a good user experience. However, the current digital home platform landscape is highly heterogeneous, with different operating systems resulting in barriers to achieve cross-platform development and testing processes for digital home applications. New envisaged applications could engage with information and services exploiting the context. However, context awareness for pervasive applications introduces new challenges for ensuring that the desired user experience is achieved. The hardware and software of the devices vary so many that it is difficult to achieve portability feature across platforms. Hence the current trend in developing interoperable applications is to use web technology instead of platform-specific APIs.

HTML5 and WebGL are fully aligned with this trend by providing the Web as a software platform for interoperable applications. They offer device orientation, geolocation management and 3D rendering, bringing from native features to web-centric development. However, constraints to render interoperable complex 3D environments are still present especially in digital home devices such as TVs,

set-top boxes, smartphones and tablets. Results described around the browser limitations to render 3D scenes of these devices, become evident the need to improve the performance of 3D applications over the digital home browsers to satisfy the prospects of the users, even if the these devices are being fitted with improved low energy-consumption GPUs.

In order to overcome this problem, the *3DMaaS* approach introduced in this paper, deploys remote servers performing the remote rendering of complex 3D scenes and then sending the frame results to a digital home device. This video streaming server approach pushes part of the graphics generation logic to the cloud and, in essence, turns the end device into a thin terminal. Driven by latency constraints, our approach proposes a hybrid solution combining remote rendering of background 3D objects, where the latency does not have a high impact on the user experience, with local browser WebGL rendering of foreground 3D objects which require low latency. Synchronization and 3D scene consistence challenges must be managed by the HTML5 application and the related complexity depends on its domain. Experiments show the results obtained by the proposed system for hybrid remote and local rendering enhance the interactive experience of 3D graphics on digital home devices proving the feasibility of interactive navigation of high complexity 3D scenes while provides a interoperable solution that can be deployed over the wide device landscape.

**Acknowledgments** - The authors would like to thank the Spanish Ministry for Industry, Tourism and Commerce, and the European FEDER funds for supporting the research activities involved in this paper.

## 3.3 Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis

- **Title:** Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis
- **Authors:** Mikel Zorrilla, Angel Martin, Iñigo Tamayo, Naiara Aginako, Igor G. Olaizola
- **Conference:** Cloud and Green Computing (CGC)
- **Publisher:** IEEE
- **Year:** 2013
- **DOI:** http://dx.doi.org/10.1109/CGC.2013.68

**Abstract** - In this paper we introduce a new platform to perform image processing algorithms over big data. The main stakeholders of media analysis are the social services which manage huge volumes of multimedia data. While social service providers have already a big resources pool of connected assets through the devices of the community, they are not exploiting them for their processing needs and they usually deploy high performance systems that run batch works. Image processing requires parallelizable atomic and lightweight tasks that can benefit from a big community of thin devices executing seamless background processes while the user enjoys other social media contents. To provide such infrastructure a client-side browser solution based on JavaScript libraries has been developed. We also describe a performance model that establishes the contexts where the solution gets ahead in terms of available resources and the processing problem nature.

### 3.3.1 Introduction

With rapid advances in multimedia production and the explosion of social media, the volume of multimedia that must be processed, analysed and tagged brings big data challenges. At the same time, dramatic decrease in the cost of commodity computing components brings large distributed computing platforms with tens or hundreds of thousands of unreliable and heterogeneous hosts. Grid systems cope with intensive processing tasks but barely face uncertainty of availability performing tasks with best-effort policy capturing all the required resources. Nowadays, Cloud Computing solutions rise to overcome elasticity in order to track processing request volume. All these solutions boost processing by deploying a monolithic, for grid, and elastic, for cloud, purpose-specific infrastructure. Future solutions must be driven by social and connectivity paradigms.

Social computing is an emerging field, which encompasses a diverse range of topics. It is mainly exploited as a vehicle for establishing and maintaining mainstream communication relationships between enterprises and customers. Major motivation of Social Business trends pursue the burgeon opportunities to monetise social knowledge, emphasising social intelligence.

However, a computing-oriented dimension for collaborative computing has not been explored yet. Social computing systems provide extra value than what is offered by computer systems alone. However, the next generation of the Internet envisages the Web as a computing rather than just publishing platform. This umbrella term has also been associated with Cloud Computing.

The rapidly increasing use of the Web as a software platform with truly interactive applications is boosted by emerging standards such as HTML5[28] is removing limitations, and transforming the Web into a real application platform middleware tackling hardware resources of the devices through JavaScript [TMAS11][TM11][ASMT11][ZMS+13].

Common multimedia processing tasks such as segmentation, clustering and classification of multimedia streams from contents stored in a repository can be easily divided and distributed in atomic tasks. We can break up data into frames that can be processed independently as frames from different scenes can be considered unrelated. This lets us divide a large stream into small chunks that a thin device can analyse comparatively quickly. In this way, servers can dispatch the work to users willing to donate their spare CPU cycles.

Our approach deploys a promising solution to cope with the big data problem behind the batch analysis of the social media managed in a social service. This paper introduces a distributed user device platform on top of a social media community. The service provider takes benefit of the huge processing capacity of its big social community to seamless perform atomic image processing tasks. To achieve it, these lightweight works are embedded by the server to the different social content accessed.

This paper provides a state-of-the-art of big data infrastructures from ongoing volunteer computing projects to web based distributed processing frameworks. We also summarise current image processing libraries exploiting browser capabilities of the user devices through JavaScript. The social distributed computing paradigm is also explained in the article giving rise to a system proposal to run multimedia analysis, making user device farm suitable for big data. A system architecture is detailed and we present performance models concluded from computational patterns studied in representative research. We introduce some image processing use cases where this infrastructure fits. Last but not least, a technical validation of our proposal is done emphasising on the performance.

### 3.3.2 Related work

Current image processing research mainly focuses on algorithm accuracy and infrastructure performance. Putting aside the first area, the analysis of huge datasets

[28] HTML5 standard specification (May 2011) http://www.w3.org/TR/html5/

of multimedia content is a typical 'big data' problem that requires massive computational resources. On the one hand, buying that amount of computational power would be incredibly expensive. On the other hand, main stakeholders of image processing solutions are the social media companies such as YouTube, Facebook or Twitter. Better and deeper tagging means increased relevance and more linked contents to engage user audience. These companies could avoid to invest in computing infrastructures, because they would have already available a huge network of user devices connected to their servers, removing the energy impact of high-performance computing systems.

Social computing research has been mainly focused on enabling technologies and specific applications guided by key directions including the emerging fields such as web science and dynamic network analysis, and the movement from social informatics to social intelligence, with an emphasis on managing and retrieving social knowledge. However, the future research lacks a core set of general scientific principles and a framework to guide social distributed processing. Putting aside the supporting technologies behind, Social Cloud research oversees a new type of computing paradigm, that inherits all the benefits provided by the conventional cloud but deployed over social community users who collectively construct a pool of resources to perform computational tasks.

SETI@home[29] is probably the main example of scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI). Users participate by running a free program that downloads and analyses radio telescope data. But from the seed of this collaborative network model, numerous initiatives[30] related to unselfish research such as astronomy, climate, astrophysics, mathematics, genetics, molecular biology and cryptography have been raised where volunteers and donors share the computing time from personal devices.

Contrary to the previous initiatives our approach does not require downloading or installation of client software, atomic lightweight tasks are instead seamless embedded in a website and performed through JavaScript engine. So users do not need to be explicitly engaged to donate spare CPU cycles.

The most similar solution is brought by Plura Processing[31]. It provides a distributed computing solution using the web to power complex processing. The

[29] SETI@home Project (May 1999) http://setiathome.berkeley.edu/

[30] List of distributed computing projects (May 2013) http://en.wikipedia.org/wiki/List_of_distributed_computing_projects

[31] Plura Processing Service (2008) http://www.pluraprocessing.com/

Plura affiliates earn revenue or drive donations by connecting computers to Plura network while their customers take benefit of one of the largest sources of computing power at affordable on-demand pricing. However, each approach focuses on different kind of problems and exploits different technologies. Moreover, our solution envisions a platform deployment on top of an already established network endorsed by a social media community.

This work [MTCK11] builds a distributed computing network based on a social graph of users you already trust assigning tasks to different nodes based on social acquaintance. It analyses several design options and trade-offs, such as scheduling algorithms, centralisation, and straggler handling. Our solution goes beyond providing the whole community resources to the service servers for specific multimedia processing tasks.

The great exponent of generic purpose massively collaborative computation with web technologies is MapReduce [CB13]. This framework for processing parallelisable problems was used by Google to completely generate Google's index of the World Wide Web. It defines a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. Available solutions forks [LMA+10] from open source Apache Hadoop[32] frameworks. This technology overcomes server-side tasks dispatching over a set of nodes. The highlight of this work [DFMGS11] is the ubiquitous nature of the image matching problem. It also analyses some image processing algorithms specifically implemented for MapReduce technology. Another image processing projects hold by this technology are: HIPI[33] [Swe11] that provides an API for performing image processing tasks in a distributed computing environment; and many more[(34)(35)(36)].
Current research goes further aggregating client-side nodes to work together with the server ones. In this direction, JSMapReduce[37] [LWB13] is an implementation of MapReduce which exploits the computing power available in the computers of the users of a web platform by giving tasks to the JavaScript engines of any web browser. The atomised nature of multimedia processing tasks tackled in our work does not fit in with the MapReduce policy oriented to server side dispatching protocols.

[32] Apache Hadoop framework (2005) http://hadoop.apache.org/
[33] Hadoop Image Processing Interface (2012) http://hipi.cs.virginia.edu/
[34] EyeALike similarity across large datasets (2012) http://www.eyealike.com/
[35] Gum Gum in-image advertising platform (2012) http://gumgum.com/
[36] Kalooga discovery service for image galleries (2012) http://www.kalooga.com/
[37] JSMapReduce JS library (2013) http://jsmapreduce.com/

Regarding JavaScript multimedia processing libraries there are an increasing number of solutions: IM.js[38] faces pixel image comparison; Resemble.js[39] deals with image analysis and comparison; Processing.js[40] addresses video and audio manipulation; Pixastic[41] manage pixel data access and manipulation of the image bringing color adjust, histogram, desaturate, edge detection, noise removal; jsfeat[42] provides image resample, equalise histogram, canny edges, fast corners feature detector, Lucas-Kanade optical flow, HAAR object detector, BBF object detector, linear Algebra module (Gaussian elimination solver, Cholesky solver, SVD decomposition, solver and pseudo-inverse, Eigen Vectors and Values) and Multiview module (Affine2D motion kernel, Homography2D motion kernel, RANSAC motion estimator, LMEDS motion estimator); ccv handles face detection and object detection[43]; nude.js[44] detects skin and body. All these features joined with JSMapReduce provide simpler and unique frontend for web developers that only must focus in JavaScript code writing and the results interpretation of multimedia analysis algorithms.

In terms of algorithm performance, historically JavaScript has been inefficient when compared to languages like C and C++, but Mozilla is recently bringing near-native application performance to the web Emscripten[45], an LLVM-to-JavaScript Compiler to translate C and C++ code into asm.js, a JavaScript subset for Firefox browser. This potential combined with the Mozilla's Web Workers solution to run scripts in background threads for Firefox browsers could provide the most powerful pure web platform for big data processing. However, it requires not only to translate the algorithms code through Emscripten but also it can only be performed on Firefox browsers. However, the browser ecosystem is heterogeneous (Chrome, Opera, Safari, IE) and the solution must fit in all of them.

Our goal is to advance current distributed computing approaches adding the social dimension and going into detail for multimedia processing purposes. To this end, our contribution in this paper is mainly twofold. First, we investigate the

[38] IM.js Image comparison pixel by pixel (2013) http://tcorral.github.io/IM.js/

[39] Resemble.js Image analysis and comparison (2013) http://huddle.github.io/Resemble.js/

[40] Processing.js multimedia processing framework (2013) http://processingjs.org/

[41] Pixastic JavaScript Image Processing Library (2009) http://www.pixastic.com/lib/

[42] jsfeat Computer Vision library (2013) https://github.com/inspirit/jsfeat

[43] ccv Computer Vision library (2013) https://github.com/liuliu/ccv

[44] nude.js Nudity detection with JavaScript and HTMLCanvas (2010) http://www.patrick-wied.at/static/nudejs/

[45] Emscripten LLVM-to-JavaScript Compiler (2013) https://github.com/kripken/emscripten

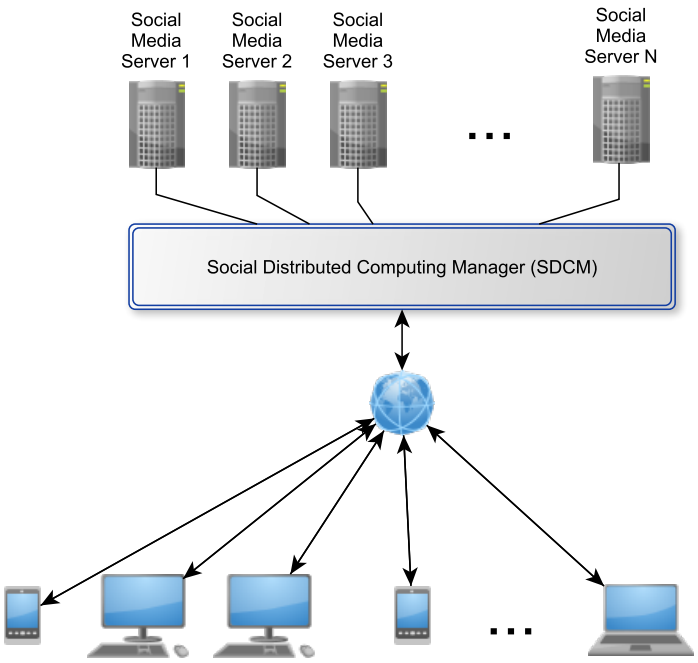


**Figure 3.8:** System Architecture

potential of the social distributed computing paradigm by introducing a design that construct distributing computing services. Second, we profile the potential of our approach to define the most suitable computationally-intensive problems regarding management overhead for task split and results splice.

### 3.3.3 System Architecture

The designed system architecture is based on a client-server architecture (Figure 3.8). It is completely oriented to have Web-based clients over HTML5, including specific JavaScript libraries. These specific purpose libraries are added inside a social media Web-based service transforming the client browser into a seamless image processing resource for the server. All the clients communicate with a main server interface through the Social Distributed Computing Manager (SDCM), which is in charge of attending them, balancing the load and distributing the requests through the different web servers available for the service.

Obviously, the different hardware and software stacks of the heterogeneous devices, span a wide spectrum of capabilities and performance. Emerging JavaScript libraries play a crucial role profiling the device capabilities to avoid impact on the user Quality of Service. Thus our approach uses minimal client resources

to avoid affecting the user's experience. Thereby, on a first step, an initial process is launched in the client in order to evaluate the client device benchmark. Hence, the server decides to assign or not a processing task. In case of suitable devices, the rated analysis algorithm complexity and the image dimensions to be processed are matched according to this score. From that moment on, the data transfer begins between the social service client and the server delivering an image or a video frame and the order to process the data. This request includes the image processing algorithm or function to run for that image, the results format, etc.

On a second step, if the server has dispatched and set up a task, the client JavaScript engine will start the background work thread seamlessly while the user continues enjoying the social media service. Once the processing task is finished, the web browser will send the obtained result to the server triggering a new processing task request to the server in order to provide idle awareness.

These fragmented work tasks will be processed by the client along the user accesses a content from the social media service. So the server is responsible for managing the task distribution and monitor if the results are correctly received. In other case the server deals with uncompleted tasks re-sending them to another clients.

The following sections present a more detailed description of the different modules of the system architecture.

#### 3.3.3.1 **Client Architecture**

The client is composed by several modules that are showed in Figure 3.9:

**Communication Layer**: This module is responsible for bridging the client-server communications, implementing the data and message protocols widely supported by HTML5 and JavaScript such as Websockets and AJAX. The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based services that need two-way communication with servers that does not rely on opening multiple HTTP connections [Fet11]. However, even if the Websocket implementation is in the roadmap of every Web Browser, nowadays there are

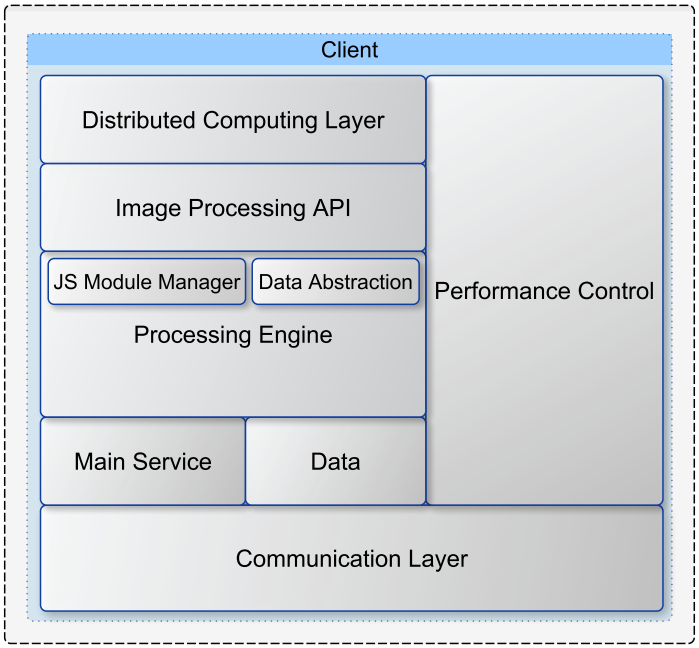


**Figure 3.9:** Client Architecture

restrictions to use it specially in mobile devices. A polling approach using Ajax technology, widely supported by any browser, is defined as an alternative to Websockets. Ajax (Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously keeping visual fluidity and behavior of the foreground Web application. Data can be retrieved using the XMLHttpRequest object [Gar05].

**Main Service**: This module is the main Web application. It is a social media Web service where the user enjoys watching media content, interacting with friends, etc. The service keeps the highest priority to provide a good Quality of Experience (QoE) to the user interacting with social media contents. However, these applications do not usually require high computing performance in the client. The following modules add background processing tasks to the browser taking benefit of the spare CPU cycles of the connected users without interfering with the experience of the user with the foreground application. This way the service can enhance the tagging of the managed contents in order to empower content relevance and discover content relations boosted by the social community engine.

**Performance Control**: This module is a transversal task which is in charge of

assessing client browser performance to distinguish client-side capabilities for extra job. If the client device is busy and cannot carry out additional work the other modules cease claiming incapable. In contrast, if the device can deal with concurrent tasks, all the modules in the client system start running and processing information.

**Data**: This module cope the transference of data between the client and the server. On the one hand, it will be very close to the Communication Layer to exchange data and messages with the server and on the other hand it will create and maintain the HTML5 Web Storage facilities for the received data. Once the local processing of the video frame is done, it will be also the responsible to adequate the results and send it to the Communication Layer to be transferred to the server. It is important to highlight that our solution is entirely run in memory by the web browser and do not access or record any client information.

**Processing Engine**: This is the core module of the client since it is the responsible to inject the downloaded image processing script, and run it with the downloaded frame. It also formats the obtained results to be sent to the server. It is based on two sub-modules:

- **JS Module Manager**: This module is in charge of the JavaScript image processing scripts injection and management. While the user is interacting with the social service, this module is the responsible of preparing and loading the scripts to be executed seamlessly without any impact in the user experience.

- **Data Abstraction**: This module adapts and formats the results of the execution of the script to be ready to send them to the server. The server will send information about how to prepare the output data so this module will format it to the adequate structure.

**Image Processing API**: This module benefits from the approach proposed in this paper by accessing to a simple Image Processing API in the server. Based on JavaScript code, it hides algorithms complexity and connects different methods to obtain specific features and they will be executed on a batch way through the different client browsers.

**Distributed Computing Layer**: This module is the orchestrator delegate running background tasks at end devices such as computers, tablets, smartphones, TVs, etc. by coordinating all the client-side modules to execute both the main

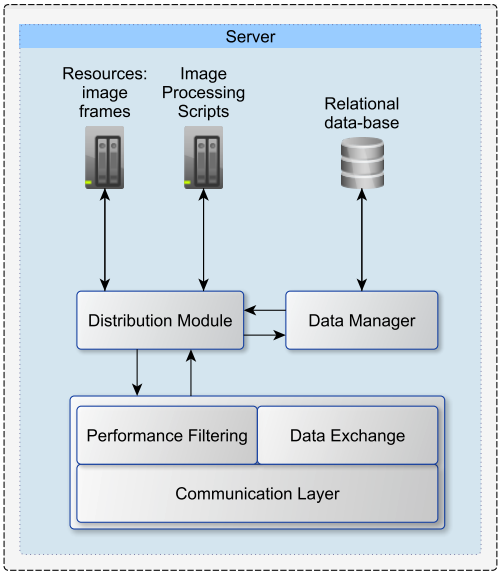


**Figure 3.10:** Server Architecture

social application and the other background processes concurrently and in a seamless way for the final user.

#### 3.3.3.2 Server Architecture

Figure 3.10 depicts the different modules of the server:

**Communication Layer**: This module manages the communications between the server and the clients. As it is mentioned in the client side, this layer is deployed on top of Websockets and AJAX communication protocols to exchange data and messages.

**Performance Filtering**: This block profiles the capabilities of each client in order to assess a specific task suitability. Each client will report its processing capability through a test done by the Performance Control to the server. According to the achieved score the server matches the different tasks depending their processing needs.

**Data Exchange**: It deals with data exchange (e.g. a image frame, scripts, etc.) through the communication layer between the client and the server.

**Distribution Module**: This is the core tasks dispatcher through the clients allocating the task, the image frame, the script and the needed information (how to format the answer, etc.). This module coordinates the communication layer, data exchange and performance filtering, but also communicates with the Data Manager and with the Resources (Image frames and image processing scripts).

**Data Manager**: This module interfaces the relational data-base which contains the information of the global tasks that should be done, in terms of what image processing scripts need to be applied to each image. This way, the Distribution Module will be able to assign a specific task relating a frame and a script following the priorities specified in the Data Manager. The relational data-base contains also information about the processing weight of the different scripts to fix to the performance capability of the client. This module is also responsible of gathering all the processing results and store them in the data-base.

### 3.3.4 Use cases

The architecture described in the previous section is quite generic purpose oriented. In order to land the concept behind to specific social computing needs, we describe here some concrete image processing topics that fit on our approach while gather the main interests for social services are:

- Face detection and recognition for face similarity and recognition across a dataset.
- Logo recognition for image content based advertising.
- Character recognition for image content based auto-tagging for social media.
- Near similarity detection for image based video copyright protection.

The aim for the different images is always to retrieve results independent from other contents, so the temporal dimension and correlation is completely removed from the distributed processing platform. Our perspective restrains problem complexity by removing the need of a subsequent multiple results consolidation. This way the social server would not need high demanding postprocessing to join client-side results that would mean processing overhead at the server-side.

Our target scenario is YouTube like social media services. This context provides a perfect environment for our system thanks to:

- Longer and continuous user sessions for videos. In contrast to text based social communities, it provides high availability of the resources with a more stable infrastructure in terms of elasticity.

- Not multi-task activities while watching videos. Users underuse their devices where background tasks can execute without interfering with the user experience.
- Residual overhead. The overhead of the data to be analysed is residual compared with the mainstream content.
- Continuous communication channel. The high transfer rate of a streaming server bridges a stable communication with lower latency.

It is very important to highlight that our solution manager anonymises the content to be analysed and the assigned tasks are entirely run in memory by the web browser and do not access or record any client information.

### 3.3.5 Performance Modeling

The performance of the proposed approach can be theoretically analysed by following the PRAM model [FW78]. According to this model, each processor has access to the shared memory and can also do computation on a local memory. However, this model has some drawbacks as it assumes that all processors work synchronously and that interprocessor communication is free. To overcome this limitations Culler et al. propose the LogP model [CKP+93]. The model is based on the following parameters:

- $L$: an upper bound on the latency or delay in communicating messages from the source to the target module.
- $o$: the overhead defined as the time that a processor is engaged in the transmission/reception of each message and cannot perform other operations.
- $g$: the gap or the minimum time interval between consecutive message transmissions/receptions at a processor.
- $P$: the number of processor/memory modules.

However, this model does not fit distinctly with the approach proposed in this paper as is it intended to deal with complex network topologies and simple processing units. Moreover, the LogP model is accurate only at the very low level hardware stack and not for practical communication systems with layers of protocols (e.g. TCP/IP).

In our case, the network topology is a star where processing nodes can dynamically be added or removed. It implies that:

- One processor acts as the central processor.
- Every other processor has a communication link with this central processor.
- Congestion may happen at the central processor (dispatcher).
- Remote processors can deal concurrently with messaging and processing tasks.

As a generalisation of PRAM and removing the overhead $o$ of LogP, Valiant [Val90] proposes the Bulk Synchronous Parallel (BSP) model. In this model, each processor can follow different threads of computation and tasks are organised as *supersteps*. A supersteps includes three stages.

1. Concurrent Computation (asynchronous)
2. Communication
3. Barrier Synchronisation

The cost of a BSP algorithm is the total cost of the sum of all supersteps, and the cost of each superstep is given by Equation 3.3:

$$C_{superstep} = \max_{i=1}^{p}(w_i) + \max_{i=1}^{p}(h_i g) + l \tag{3.1}$$

$$C_T = \sum_{s=1}^{S} W_s + g \sum_{s=1}^{S} H_s + Sl \tag{3.2}$$

$$W = \max_{i=1}^{p}(w_i), H = \max_{i=1}^{p}(h_i) \tag{3.3}$$

where:

- $p$ = number of processors
- $S$ = number of supersteps
- $l$ = synchronisation periodicity
- $g$ = communication cost
- $h$ = maximum number of incoming or outgoing messages per processor

- $w$ = computation time

In our case, the communication channel is granted through the already established video streaming flows. It ensures good communication performance for relatively small data communications. When the required bandwidth is comparable to the video transmission bitrate, it may alter the QoS and $g$ would increase.

The synchronisation periodicity could be removed for many use cases where the Social Distributed Computing Manager (SDCM) creates independent tasks. For example, to process all the instances of a database can be considered as a big single superstep that synchronises when all the databased is processed. In this case, we could model the computational cost as:

$$C_T = W_s + gH_s + l = \max_{i=1}^{p}(w_i) + \max_{i=1}^{p}(h_i g) + l \tag{3.4}$$

However this assumption does not allow the fact that fastest processors can start a new task once they finish the previous one. To do that, we define the model from the process perspective. According to this view, we can consider the total cost as the time needed by the network of resources to process each independent task by means of average communication and processing costs (Equation 3.5).

$$C_T = \frac{\hat{W}_s}{p_u} + \frac{\hat{g}\hat{H}_s}{p_t} + Sl \tag{3.5}$$

In order to estimate the benefit of the presented approach compared with local processing, we can apply the same cost model (Equation 3.5) to a local multicore processor. In this case, the data access will be much faster but the number of processors might be much lower.

In order to give a comparative estimation, we will consider 4 types of devices with different connectivity and processing power (Table 3.5).

$H_s$ depends on the algorithm implementation, not on the device or communication infrastructures. So it is closely related to the defined use cases. Tasks are considered as batch processes, thus, it will not be contemplated in the cost comparison. The $Sl$ factor will be reflected as a internal management factor that will require some computational power at the server side to be consolidated. However, it will not incur in extra time cost per operation as local parallel processing

[46] http://www.tomshardware.com/reviews/fx-8150-zambezi-bulldozer-990fx,3043-14.html

[47] http://www.legitreviews.com/article/1988/2/

**Table 3.1:** Estimated processing and communication properties for different types of devices.

| ID | Device | Connectivity | Average Bandwidth | Average GFLOPS |
|---|---|---|---|---|
| (m) | Mobile phone | UMTS | 3Mbit/s | 0.05 |
| (t) | Tablet | Wifi | 8Mbit/s | 0.08 [46] |
| (p) | PC | DSL | 20Mbit/s | 2.5 |
| (s) | Server | SATA | 6Gbit/s | $p_s$x82.8 [47] |

at the client side. Keeping in mind that we are in a streaming session context, initial delays can be disregarded as the data needed to communicate the server and the remote processors are directly added to the ongoing stream. The following equations depict a breakdown of the cost estimation under the conditions specified by Table 3.5. $f_{xy}$ represent the utilisation factor of the resources that will be available to process the tasks. In order to avoid any annoyance in the user experience, $f$ will be set to 0.15 both for bandwidth and for processing power. The cost to establish a new thread and its management its denoted by $\hat{m}$. It is a fixed estimated value and mainly considers the cost on the server side.

Equation 3.6 represents the sum of the partial costs of the different type of end devices (mobile, tablet and PC) while equation 3.7 models the computational cost of each individual group of devices with similar characteristics.

$$C_T = \left( \sum_{i}^{n} \frac{1}{C_i} \right)^{-1} \tag{3.6}$$

$$C_i = \frac{\sum W_i}{f_{pi} \cdot F_i \cdot p_i} + \frac{\sum g_i}{f_{bi} \cdot \hat{b}_i \cdot p_i} + \hat{m} \cdot p_i \tag{3.7}$$

Following the same model, the cost of a multicore server with a single internal shared memory is:

$$C_s = \frac{W}{f_{ps} \cdot F_s \cdot p_s} + \frac{g}{f_{bs} \cdot \frac{\hat{b}_s}{p_s}} + \hat{m} \cdot p_s \tag{3.8}$$

In order to compare a distributed computing approach with a dedicated local server, different $C_T$ and $C_s$ have been calculated. Figure 3.11 shows different cases of performance behaviour for several values of model parameters. A lineal increment of processors is compared for different sizes of $W$, $g$ and $\hat{m}$. As it can be observed, the maximum benefit of our proposed social computing network is

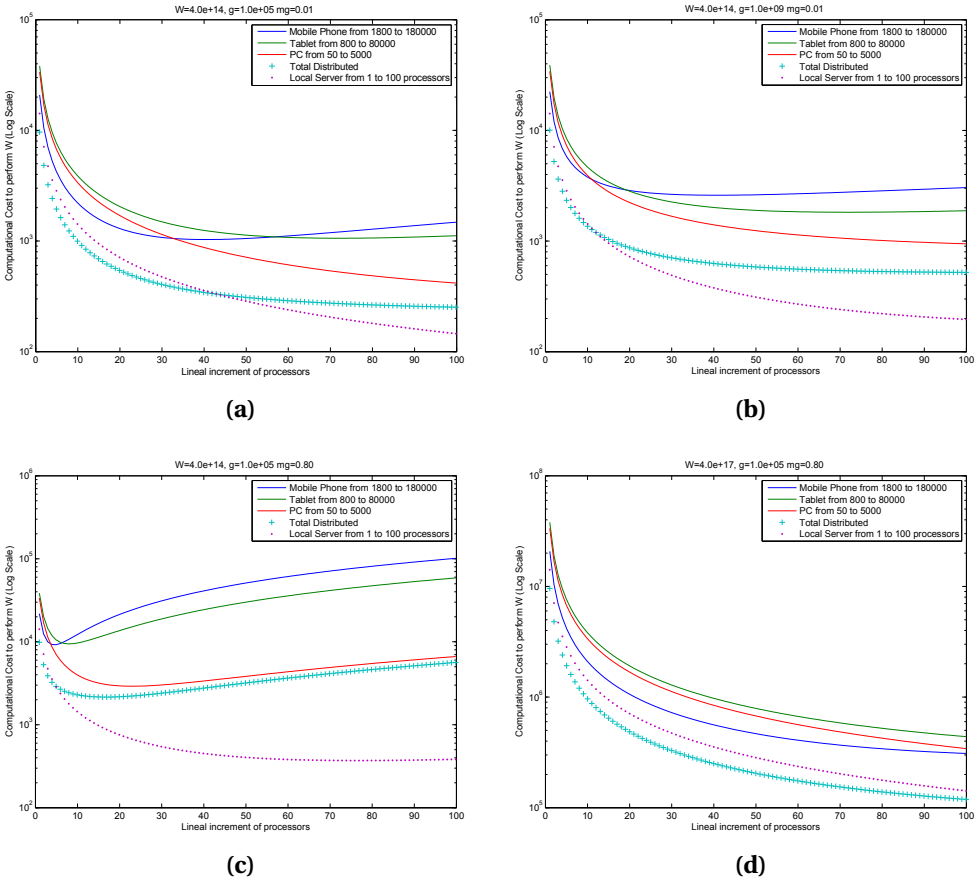


**Figure 3.11:** Computational cost estimation under different sizes of work load ($W$) communication cost ($g$) and thread management cost ($\hat{m}$) .

obtained for those use cases with higher computational load. The communication costs can be the main bottlenecks unless bandwidth conditions or utilisation factor are increased. Figure 3.11c shows that the efficient management of all the created threads becomes a critical factor as well.

### 3.3.6 Validation

In order to provide some reference experimental values for the parameters involved in the theoretical performance modeling in a specific context, we have compared the performance of a distributed computing architecture with a dedicated local server approach. We have implemented the same testbed for both scenarios: processing of an image using JSFeat libraries in distributed devices and the OpenCV libraries for the standalone local server execution. The testing image resolution was 1200x1600 and compressed in JPEG. The processing method

consisted on applying a grayscale converter and a Canny detector due to the necessity of finding reciprocal functions in both libraries to provide a homogeneous framework.

As validation parameters two elapsed times have been measured, the global processing time, which includes streaming of the image to be analysed and the actual algorithm processing, and the computation time that just includes the algorithm runtime. We have calculated the mean and the variance of 100 task samples. For distributed approach, client-side devices we used a computer, a tablet and a smartphone. As can be seen in Table (3.2), most of the global processing time is used for algorithm computing, justifying the capacity of this kind of architectures to minimise data streaming time consumption in relation to global processing time.

In conclusion, obtained results are coherent with the assumptions taken in the theoretical model and the relations of the different parameters involved. Even more, comparison of these results with the outcome of a standalone server execution (see Table 3.3) underline the opportunity of using web based distributed platforms as a solution for large scale processing.

**Table 3.2:** Mean and variance time consumption for client-side devices in a distributed architecture.

| **Architecture** Client-side (JavaScript) | **Global Process** $\mu$ (ms.) | **Computation** $\mu$ (ms.) | **Global Process** $\sigma^2$ | **Computation** $\sigma^2$ |
|---|---|---|---|---|
| CPU 3 dual core | 258 | 240 | 42 | 18 |
| Asus Eepad transformer | 2710 | 1744 | 97 | 73 |
| Nexus 4 | 1694 | 1537 | 156 | 130 |

**Table 3.3:** Mean and variance time consumption for local stand-alone processing.

| **Architecture** Server-side (OpenCV) | **Global Process** $\mu$ (ms.) | **Computation** $\mu$ (ms.) | **Global Process** $\sigma^2$ | **Computation** $\sigma^2$ |
|---|---|---|---|---|
| CPU 2.3 dual core | 277 | 101 | 24 | 37 |

### 3.3.7 Conclusions

The social distributed computing platform leverages the computing power of potentially thousands of devices to come up with a cheap, flexible solution for social media analysis. Users contribute a fraction of their computing time. Despite the lower efficiency, compared to stand-alone and Grid solutions, the barrier to entry is low thanks to the potential larger cluster to reach an astounding number of machines for social services. This context enables us to solve big data problems previously unachievable. To be more specific the main stakeholders of image processing, the social services, can apply background work for batch analysis over the whole social media dataset.

However, the infrastructure deployment must keep in mind requirements and constraints of social services. On the one hand, there is a major requirement attached to keep the Quality of Service means to avoid draining the users' bandwidth and processing power. On the other hand, the solution must deal with resources availability due to the high elasticity related to the spontaneous presence nature of users. These conditions settle the framework of the image processing work which implies the necessity of atomic and lightweight image processing tasks.

The proposed system spreads over a client-server architecture working totally over Web-based clients. Emerging JavaScript technology enables to manage and run seamless background tasks while the user interacts and enjoys a social media service without affecting the Quality of the Experience. The approach provides a community of devices as a resource for computing tasks and there is no need to install or develop client applications but adding a distributed computing layer to the HTML-based main service.

Regarding a performance modeling over an adapted BSP model, the maximum benefit of the proposed social distributed computing platform is obtained for use cases with high computational load. This fits with the social media service providers needs, that often require complex image analysis processes for a huge volume of data. The target scenario is a social media content service (like YouTube) providing a favorable scenario. It brings beneficial features such as continuous communication channel, residual overhead compared with the video itself, not multi-task user activity, device underuse and longer sessions that foster high availability of resources for the distributed computing tasks.

In addition, validation experiments back up the theoretical performance modeling and remark the opportunity of using the Web-based social distributed

computing solution for large scale processing in comparison with the outcome of a server grid approach.

To conclude, the proposed system deploys a promising solution to cope with the big data problem that the social media service providers deal with, through a social distributed computing platform. Service providers benefit from the huge processing capacity of the social community adding to the main service, via web browser, seamless background processing tasks for image analysis.

## 3.4 SaW: Video Analysis in Social Media with Web-based Mobile Grid Computing

- **Title:** SaW: Video Analysis in Social Media with Web-based Mobile Grid Computing
- **Authors:** Mikel Zorrilla, Julián Flórez, Alberto Lafuente, Angel Martin, Igor G. Olaizola, Iñigo Tamayo
- **Journal:** Transactions on Mobile Computing
- **Publisher:** IEEE
- **Status:** Submitted in 2016

**Abstract** - The burgeoning capabilities of Web browsers to exploit full-featured devices can turn the huge pool of social connected users into a powerful network of processing assets. HTML5 and JavaScript stacks support the deployment of social client-side processing infrastructure, while WebGL and WebCL fill the gap to gain full GPU and multi-CPU performance. Mobile Grid and Mobile Cloud Computing solutions leverage smart devices to relieve the processing tasks to be performed by the service infrastructure. Motivated to gain cost-efficiency, a social network service provider can outsource the video analysis to elements of a mobile grid as an infrastructure to complement an elastic cloud service. As long as users access to videos, batch image analysis tasks are dispatched from the server, executed in the background of the client-side hardware, and finally, results are consolidated by the server. This paper proposes SaW (Social at Work) to provide a pure Web-based solution as a mobile grid to complement a cloud media service for image analysis on videos.

**Index Terms** - Distributed computing, image analysis, multimedia databases, multimedia systems, social media, web-based architecture

### 3.4.1 Introduction

The social media paradigm has led to a significant rise in the volume of user generated content managed by social networks with millions of users accessing services, each of them often using multiple devices at the same time. Service providers aim to engage audience, eager for contents, by boosting the media relevance. To this end, a deeper automatic tagging enables better matching of user interests with the content database and reveals underlying connections between items, such as applying face detection mechanisms or content-based indexing to find related videos. Image analysis algorithms empower automatic retrieval of salience features but they also involve computing-intensive functions. Therefore, the processing requirements grow substantially when all the media items comprising the social network database are analysed. Here, on the one hand big data challenges arise when social services have continuously increasing databases, while on the other hand more and more processing resources are required to analyse all the content.

Grid and Cloud technologies provide High Performance Computing systems that aim to satisfy these requirements. However, as pointed in [NBRK11], other under-explored alternatives could enhance the trade-off between infrastructure cost, elapsed time and energy saving. It would depend on the number of available processing nodes, the inherent characteristics of the tasks to be performed in parallel and the data volume.

To deal with the aforementioned context, this paper introduces a new concept of Social at Work: SaW. It aims to complement a Web-based social media service with all the idle devices, mostly mobiles, that usually have underexploited resources while accessing the service. SaW goes beyond the Infrastructure as a Service (IaaS) model, creating a system related to Mobile Grid Computing [AM06] concept with the available CPU and GPU resources of the different client devices to complement a virtualised cloud server, which provides the social media service.

Inspired by the Mobile Grid Computing and the Mobile Cloud Computing (MCC) [HXW13] research fields over a social network mainly based on video content, SaW aims to bring together the huge pool of users permanently connected to media services in social networks and the ever increasing processing capabilities of most of their devices. As a consequence, service providers will embrace the community assets building a device centric grid to improve the social service by means of media analysis. Thus, Social at Work (SaW) concept enables service

provider to recruit spare CPU/GPU cycles of client devices into an active gear of the social platform, saving cloud resources to the server when the connected clients can perform those tasks.

To achieve a SaW system, some remaining issues must be faced such as turning a Web client into a runtime application framework, shrinking Web engines performance gap between native applications and Web-apps, deploying communication layer to distribute background analysis and tracking processing request volume dealing with uncertainty of resources availability and elasticity.

First, the current device ecosystem is highly heterogeneous, with different operating systems and programming languages, resulting in complex software cross-platform development. Here, SaW proposes a pure Web-based approach since Web technologies overcome the interoperability barriers. HTML5 turns the Web into a real application platform middleware accessing hardware resources of the appliances through JavaScript [ASMT11].

Second, in order to exploit native GPU and multi-CPU potential of a device, WebGL and WebCL bindings to OpenGL and OpenCL run hardware-accelerated, parallel and cross-platform programs. So, they endow Web applications with parallel computing capabilities, accelerating Web applications for intensive image processing [JBG12].

Then, Ajax (Asynchronous JavaScript and XML) [Gar05] and Websockets [Fet11] are employed as a vehicle for establishing and maintaining mainstream communication between server and clients, transforming the classical synchronous request-response model into a full bidirectional one. This feature enables the server to send asynchronously updates to the client-side browser and to deliver background data.

Finally, the possibility to perform image processing tasks in parallel, such as feature extraction, segmentation, clustering and classification, eases to leap scalability. Due to the video stream nature, composed by individual frames, they can be easily split into independent tasks ready to be distributed. Beyond, the intrinsic presence of key frames in video coding, makes easier navigation and selection of representative images. Servers can dispatch the tasks to users' devices where they are run in the background. These background Web browser applications must balance the mechanism to leverage all available computing resources while provide the best possible user experience.

Thus, the validation of the approach can be explained in terms of the net benefit obtained in the server by delegating part of the tasks. The equation includes

two relevant keys, which are based on certain parameters that are dependent on the technological state-of-the-art and the users' social media consumption habits: (1) the amount of work that can be distributed to the client devices, which depends on the number of available clients, their capabilities, and the fraction of resources they can dedicate to background tasks without disturbing the user experience, and (2) the extra work created in the server to manage the task scheduling, which should be residual in comparison with the saved work or at least not exceed it.

#### 3.4.1.1 **Contributions**

This paper proposes a SaW system for media analysis of the content collection in a social service. Drawing inspiration from volunteer computing initiatives for big data, this paper presents a pure Web-based distributed solution. SaW is deployed on top of the user appliances of a social media community, including the hardware-accelerated features for suitable devices. Thereby, the service provider gains the immense processing ability of its big social community to perform independent background hardware-accelerated image processing tasks. To achieve it, these queued tasks are embedded to the different social media services accessed by the users.

More specifically, the main contribution can be translated to two different topics. On the one hand, we propose a pure Web-based architectural design of the SaW concept enabling an interoperable solution. On the other hand, the article provides a proof-of-concept implementation of SaW using WebGL and WebCL technologies, in order to evaluate the SaW approach supported by experimental results and an analysis of the performance based on a previous model extending it by terms of GPU usage.

#### 3.4.1.2 **Paper structure**

This paper starts with the related work in Section 3.4.2, exploring the different Internet-based computing models and analysing their existing mechanisms for the interoperability, task distribution, support for parallel processing and different data structures. Section 3.4.3 presents the main contribution of the paper with the definition of the SaW concept. It depicts the contributions of SaW to the aforementioned related work, presents a suitable scenario for SaW on a social media service, defines the design objectives of the SaW architecture, presents

a pure Web-based architectural design, and analyses the considerations of the architecture regarding the defined design objectives.

It follows with the evaluation of the SaW approach in Section 3.4.4. Subsection 3.4.4.1 describes the experimental results in terms of scalability over a proof-of-concept implementation of SaW using WebGL and WebCL, subsection 3.4.4.2 presents a performance analysis, extending the already published model in [ZMT+13] in terms of GPU, and in subsection 3.4.4.3 some remarks regarding the validation of the SaW hypothesis are presented. Finally, Section 3.4.5 presents the conclusions.

## 3.4.2 Related work

This section presents the related work, providing a definition of the Internet-based computing models and focusing on the different topics addressed by distributed computing: the interoperability, the task distribution managing, the parallel processing capabilities and the different data structures.

### 3.4.2.1 Computing Models

This section describes the main involved concepts in terms of Internet-based computing, where shared resources, data and information are provided to computers to reach a common goal.

**Grid Computing**

Grid Computing [FKT01] has been an important paradigm in distributed systems for the last two decades. Basically, a grid is a network system where computing tasks are distributed to use non-dedicated computing resources, which may include servers or client computers. The high potential of the nowadays abundant and frequently idle client hardware boosts the opportunistic and delay-tolerant [CK10] use of client resources in the grid. In this volunteer computing SETI@home is the most popular example. SETI@home [set99] approach has been the pioneer of big data grid infrastructures taking benefit of Internet-connected computers of volunteers. SETI@home has spread the collaborative network model to other unselfish research in areas such as astronomy, climate, astrophysics, mathematics, genetics, molecular biology and cryptography where volunteers and donors share the computing time from personal devices.

**Mobile Grid Computing**

Grid Computing is characterised by the heterogeneity of the resources in both amount and nature, by the sporadic availability, churn and unreliability of the devices, and by their anonymity and lack of trust. These issues are more relevant in Mobile Grid Computing (MGC) [AM06], where computing resources include mobile devices with wireless communications, and therefore prone to disconnections and other eventualities.

**Cloud Computing**

More recently, Cloud Computing [AFG+10], a new paradigm of distributed computing where virtualised computing resources are provided on-demand, has experienced a dramatic growth. Nowadays the cloud is a cost-saving opportunity for many enterprises [Liu13] and many cloud vendors [WJW13]. Amazon is a popular cloud service provider with solutions like Amazon Simple Storage Service S3 and the Elastic Cloud Computing EC2 as an interface to them. Eucalyptus [NWG+09] is an open source cloud implementation on top of Amazon EC2.

Being not tied to a specific hardware model, Cloud Computing enables an improved time-to-market for services achieving: a reduced infrastructure deployment time thanks to an increased service availability and reliability; rapid creation of additional service instances; and cloud interoperability, which lets professionals deploy a service on multiple clouds. Thus, cloud computing provides theoretically unlimited scalability and optimised service performance.

Since the costs of cloud solutions are a key factor, new models are required to fit better with specific applications, infrastructure environments and business contexts. These new models are classified in three, according to the different virtualisation layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). An example of how client devices can be integrated into cloud services is STACEE [NBRK11], which proposes a peer-to-peer (P2P) cloud storage where different devices can contribute with storage to the cloud.

**Mobile Cloud Computing**

In Mobile Cloud Computing (MCC) [HXW13] computing resources include mobile devices as clients of the virtualised services, usually following the classical client-server asymmetric model, which involves a one way communication direction produced by requests from mobile clients to cloud services. Nevertheless symmetric MCC models have been also proposed [NBRK11], where mobile devices populate a cloud offloading the tasks to be performed by the service infrastructure. This pushes a user-centric strategy to MCC solution shifting to new

models: Mobile as a Service Consumer (MaaSC), Mobile as a Service Provider (MaaSP), and Mobile as a Service Broker (MaaSB).

MaaSC model is inherited from the traditional client-server design where mobile devices are mere service consumers. Here, mobile devices outsource their computation and storage functions onto the cloud. MaaSP switches the role of the device from a service consumer to a service provider. Last but not least, MaaSB can be considered as an extension of MaaSP, where the device gateways other handhelds or sensing nodes. Moreover, the proxy mobile device can also provide security and privacy protections to the data.

MaaSC is the most common MCC service model. Most of the MaaSC solutions, such as CloneCloud, MAUI, ThinkAir, Dropbox, GoogleDrive provide computation task offloading service for mobile devices, keeping the mobile device thin. However, with the recent advances, the features of handheld device are getting closer to regular laptops catalysing new opportunities for MaaSP deployments, as it is the case of STACEE.

#### 3.4.2.2 **Interoperability**

The interoperability in heterogeneous networks implies two abstraction levels. First, the deployment of solutions over specific architectures and operating systems. Then, the interfaces for remote operation and orchestration over a distributed system.

Despite the wide support of SETI@home, HTCondor or Eucaliptus to different architectures and operating systems, including GNU/Linux, Windows and some of the Mac OS platforms, the main drawbacks of these solutions lay on the heterogeneous computing on a variety of modern CPUs, GPUs, DSPs, and other microprocessor designs. The trend towards heterogeneous computing and highly parallel architectures has created a strong need for software development infrastructure in the form of parallel programming languages and subroutine libraries supporting heterogeneous computing on hardware platforms produced by multiple vendors [SGS10]. In response to this completely new landscape, OpenCL [ope12] is a new industry standard adopted by Intel, AMD, Nvidia, Altera, Samsung, Qualcomm and ARM holdings.

Service interoperability between different cloud providers requires standard interfaces and formats for managing virtual appliances. Nowadays, due to the lack of standard way for cloud managing, each provider publishes its own APIs.

In order to establish a universal connection, some proposals have been released [MVML13]:

- OCCI [OCC] defines a protocol and API specification for remotely managing of cloud computing infrastructures,
- CIMI [CIM] targets to set an interface and a logical model for managing resources within a cloud, and
- CDMI [CDM] establishes an interface for manipulating data elements from the cloud.

OpenNebula [ope] and Eucalyptus have made important contributions in the deployment of interoperable cloud platforms. OpenNebula implements the OCCI and CDMI specifications to enable interoperability among heterogeneous cloud platforms, whereas Eucalyptus incorporates different well-known interfaces using Amazon Web Services (AWS) [ama] as a de facto standard.

The rapidly increasing use of the Web as a software platform [ASMT11] with truly interactive applications is boosted by emerging standards such as HTML5 and WebGL that are removing limitations, and transforming the Web into a real application platform middleware to address the interoperability problem. HTML5 applications can be packed for the different execution environments providing interoperability with minor changes through independent OSs. That is why HTML5 is being strongly promoted by the standardisation bodies and a sector of the market to achieve a HTML5 marketplace instead of the available proprietary ones, such as Android Market, iOS App Store, etc. All the previously described technologies put aside new breakthroughs that turn the Web into a real interoperable application framework over the heterogeneous mobile platforms.

In this line, the ComputePool component of the Nebula cloud provides computation resources through a set of volunteer compute nodes [CWH13]. Compute nodes within a ComputePool are scheduled by a ComputePool master that coordinates their execution. The task is executed on a compute node inside a Google Chrome Web browser-based native client sandbox. Thus it provides a secure way to access local user device computational resources inheriting Web security policies to avoid compromising users' local data.

#### 3.4.2.3 Task Distribution

In our application area, social media analysis, the batch processing to be executed can be easily split into independent tasks ready to be distributed. This way, servers can dispatch the work to different processing nodes.

Focusing on generic purpose massively collaborative computation with Web technologies, MapReduce [CB13] has a noticeable position. It has been employed by Google to generate its search engine's index of the World Wide Web. In [LMA+10] another solution is proposed to overcome server-side task dispatching over a set of nodes, based on open source Apache Hadoop [had05] frameworks. The work proposed in [DFMGS11] highlights the ubiquitous nature of the image matching problems analysing some image processing algorithms specifically implemented for MapReduce technology. Another image processing projects hold by this technology are: HIPI [hip12] [Swe11] that provides an API for performing image processing tasks in a distributed computing environment; and many more [Eye12] [gum12] [Kal12]. Current research goes further aggregating client-side nodes to work together with the server ones. In this direction, JSMapReduce [jsm13] [LWB13] is an implementation of MapReduce which exploits the computing power available in the computers of the users of a Web platform by giving tasks to the JavaScript engines of any Web browser. JSMapReduce provides simple and unique frontend for Web developers that only have to focus in JavaScript code.

MapReduce defines a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. An alternative to transform the Web browser into a distributed computer middleware [CPK+13] can be also created on top of Node.js [TV10]. It provides more freedom to meet new requirements, to keep full code control and to ease third parties integration.

Nebula [CWH13], which uses volunteer edge resources for both computation and data storage, assigns tasks based on application-specific computation requirements and data location. Nebula also implements numerous services and optimisations to address these challenges, including location-aware data and computation placement, replication, and recovery. Nebula considers network bandwidth along with resources computation capabilities in the volunteer platform. Consequently, resource management decisions optimise computation time as well as data movement costs. In particular, computational resources can be selected based on their locality and proximity to the input data, whereas data

might be staged closer to efficient computational resources. In addition, Nebula implements replication and task re-execution to provide fault tolerance.

Finally, concerning who launches the requests for task distributions, two approaches are possible: a push model where the service delegates a set of tasks over a set of available resources, and a pull model where idle computing nodes request for new jobs to be performed.

#### 3.4.2.4 **Parallel Processing**

The definition of smaller tasks could bring finer granularity easing efficient processing strategies based on parallel execution in some scenarios. Hence, independent job scheduling can produce significantly better performance. Here, the social media nature brings some computational benefits. First, media can be easily decomposed in independent frames or clips. Second, the possibility to perform tasks in parallel of the multimedia processing algorithms such as segmentation, clustering and classification eases to leap the scalability dimension. Third, the multimedia processing work fits with continuous advances on parallel processing of multimedia data over GPU architectures.

With the emerging hardware acceleration technologies to exploit GPU and multi-core architectures, the parallel programming languages and the hardware computing platforms are getting closer. The most representative languages that aim to enable dramatic increases in computing performance by harnessing the power of the GPU are [YHL11]: CUDA [cud07] for NVIDIA devices provides a general purpose scalable parallel programming model for writing highly parallel algorithms; OpenMP [ope13] has established a method and language extension for programming shared-memory parallel computers. OpenMP, combined with MPI [mpi96] specification for message passing operations, is currently the de-facto standard for developing high-performance computing applications on distributed memory architecture. The underlying mechanism consists of partitioning loop iterations according to the performance weighting of multi-core nodes in a cluster. Another mainstream options are pthreads, Cilk, Ct/RapidMind/ArBB, TBB and Boost threads [JLN12]. These solutions remove barriers by providing abstraction layers for thread block managing, shared memory handling and synchronisation scaling.

MaaSP solutions must meet heterogeneity of browser ecosystem (Chrome, Firefox, Opera, Safari, Edge, IE). The SaW system targets all of them being able to

exploit underlying hardware. The cross-entry point is bridged by WebGL and WebCL. In essence, WebGL allows communication between JavaScript applications and the OpenGL software libraries, which access the host's graphics processor. Thereby, it enables use of the hardware's full capabilities not only to perform advanced 3D objects and effects rendering but also for general purpose algorithms, such as image processing. WebCL is designed to enable Web applications with high performance and general purpose parallel processing on multi-core/many-core platforms with heterogeneous processing elements. It provides ease of development, application portability, platform independence, and efficient access through a standards-compliant solution [JBG12]. Thus, WebGL excels in graphics applications while WebCL fares better when more flexibility is required in execution platform selection, load balancing, data formats, control flow, or memory access patterns [AKA+12].

An implementation example is CrowdCL [MC13]. It presents an open source framework for volunteer computing with OpenCL applications on the Web.

#### 3.4.2.5 **Data Structures**

The scale and diversity of big data problems has inspired many innovations in recent years. Different alternatives to Relational Database Management Systems (RDBMS) have emerged to fit different big data applications.

Not only Structured Query Language (NoSQL) systems, are rapidly gaining popularity and market traction overcoming limitations of relational databases [TBSD13]. The NoSQL databases were designed to offer high performance, in terms of speed and size, with a trade-off of full ACID (Atomic, Consistent, Isolated, Durable) features [HHLD11]. These storing systems include commercial solutions such as Amazon DynamoDB, Google BigTable, and Yahoo PNUTS, as well as open source ones such as: Cassandra, used by Twitter, Facebook and some other corporations; HBase, as part of the Hadoop project; and MongoDB. All of them focus on scalability and elasticity on commodity hardware. Such platforms are particularly attractive for applications that perform relatively simple operations (create, read, update, and delete). They combine low-latency features with scaling capabilities to large sizes querying engine schedules and optimizing its execution.

NoSQL data stores offer various forms of data structures such as document, graph, row-column, and key-value pair enabling programmers to model the data closer to the format as used in their application.

### 3.4.3 SaW: Social at Work

Influenced by the underlying concepts and technologies, SaW system deploys an opportunistic and delay-tolerant distributed computing platform queuing media analysis tasks over a set of trusted devices. As said before, cloud services imply a cost-saving opportunity to service providers, but depending the requirements of the service, it could still be highly demanding. This kind of services usually have a huge pool of users permanently connected to it. Moreover, second screen and multi-device media experiences are becoming very popular [NAP14a] [Nie14d] [Nie14c]. In the social media scenarios considered in this paper, users access them usually from mobile devices, which have increasing processing capabilities that are mostly idle. This pushes service providers to go deeper in the cost-saving opportunity using the mobiles as an infrastructure, replacing partially cloud resources. Regarding the computing model, SaW extends the cloud computing Infrastructure as a Service (IaaS) concept to the MCC paradigm, coining a new term of Mobile as an Infrastructure Provider (MaaIP) working together with a cloud service, in a kind of MaaSP. In SaW requests match a two-way communication pattern, since servers request clients to hire resources from mobile devices. However, the different scenarios to be performed on top of the SaW system do not require a symmetric model.

To address the heterogeneity of infrastructure, and sharing Nebula design, SaW system does not require to download or install any software in the client-side thanks to a fully Web-browser based execution stack. SaW goes beyond Nebula's ComputeTool performance by emphasising the Web stack that foster hardware-accelerated parallel programming for GPU and multi-CPU over the Web browser.

Regarding task distribution, in order to address the design objectives of elasticity, performance and security, SaW server-side schedules the queued tasks meeting computation requirements and processing availability of the client devices. This asynchronous execution model ensures control to avoid duplicities for a same task, but it does not provide support for intertask communication. Moreover, to exploit parallel processing capabilities in client devices, SaW brings hardware-accelerated performance through the JavaScript engine, WebGL and WebCL, leveraging full GPU and multi-CPU potential.

Finally, related to the data structures, SaW takes advantage of the document-oriented NoSQL technologies for media repositories fitting into one-to-many relationships of a social service.

#### 3.4.3.1 SaW Use Case

Service providers aim to engage audience, eager for contents, by boosting the media relevance. Therefore, it is necessary to improve the matching of user interests with the huge content database, and reveal hidden connections between items through a deeper tagging. In other words, the service is enhanced by improving the media content indexing.

The target scenario of SaW is a Web-based social media content service, such as YouTube[48] or Vimeo[49]. This target scenario brings beneficial features to the SaW system. First, this scenario provides a continuous communication channel, since users are typically consuming video content for some minutes without interruption, with an active application that provides the content through an adequate bandwidth. Users are aware of the required bandwidth to watch media content so they will try to select a high speed network or an appropriate coverage of 3G/4G mobile network. On second place, the SaW approach introduces a residual bandwidth overhead comparing with the video itself on this scenario. SaW will add an extra frame with a processing code to the connection, but it will be residual comparing to the data volume of a progressive download or streaming of a video. On third place, users do not usually perform any other task on the device while consuming video content. This ends to an underusing device with idle resources.

Finally, the use of an additional screen (e.g., a smartphone) accessing related content while consuming the mainstream video on a first screen (e.g., a TV set) [ZBD+15] boosts a very favourable scenario for the SaW approach, since a single user provides multiple idle devices at the same time connected to a single service.

#### 3.4.3.2 SaW Design Objectives

SaW targets a MaaIP scenario where a mobile device provides a computing component within a cloud resource system. This concept holds a key driving force moving the provisioning of processing core assets to harvesting huge amounts of available devices. Nevertheless, MaaIP, as an extension of MaaSP and Mobile Grid Computing, opens some challenges such as elasticity, performance, security and privacy, that are design objectives for the SaW architecture proposed in this article.

**Elasticity**

[48] YouTube Web site: https://www.youtube.com
[49] Vimeo Web site: https://vimeo.com

In terms of service elasticity, it is mandatory to gain cloud ability to automatically scale services and infrastructures for cost reduction when infrastructure and platform sizes are adapted to service demands. This needs of rapid and dynamic provisioning mechanisms to provide efficient service virtualisation. This factor is even more critical in SaW, when mobiles devices come into action as an available infrastructure which is unstable, with a discontinuous connectivity and with specific features such as limited battery autonomy. This means dealing with uncertainty of the availability of the resources managed by a notification mechanism providing presence awareness and performance information. This aspect turns task independence into a major condition.

**Performance**

The objective is not only to analyse all the dataset but to perform it efficiently and in a cost-saving way. In big data a residual inefficiency is multiplied by the dataset dimension with severe impact on the global system. This means scheduling and dispatching mechanisms must be implemented to orchestrate all the elements keeping efficiency for data transmission overhead related to work delivery. Furthermore, it is important to match processing needs with device capabilities and minimise re-execution of uncompleted tasks. Thus, the Web client must perform a benchmarking test in order to assess the processing capabilities and the hardware assets disposal of the user's device

**Security and Privacy keys**

Security and privacy are major concerns for cloud infrastructures even when data is hosted on a corporative data center. However, it turns into a severe issue once the data leaves the corporative firewall. The media managed in social networks consists of images, audio and video elements shared with friends. Such information is highly privacy-sensitive, and malicious attackers may access a target user's obtaining private information. Additional issues comes when dealing with security and privacy of the node provider, the owner of the device. However, to meet both dimensions, content an device owner, a combination of policies should be applied over the data transmission and storage.

#### 3.4.3.3 SaW Architecture

The deployed SaW solution works over a client-server architecture (see Figure 3.12). It improves the architecture presented on [ZMT+13] towards a hardware accelerated approach, considering all the new aspects introduced by usage of GPU resources within SaW concept. On the server-side there is a SaW Scalable Cloud

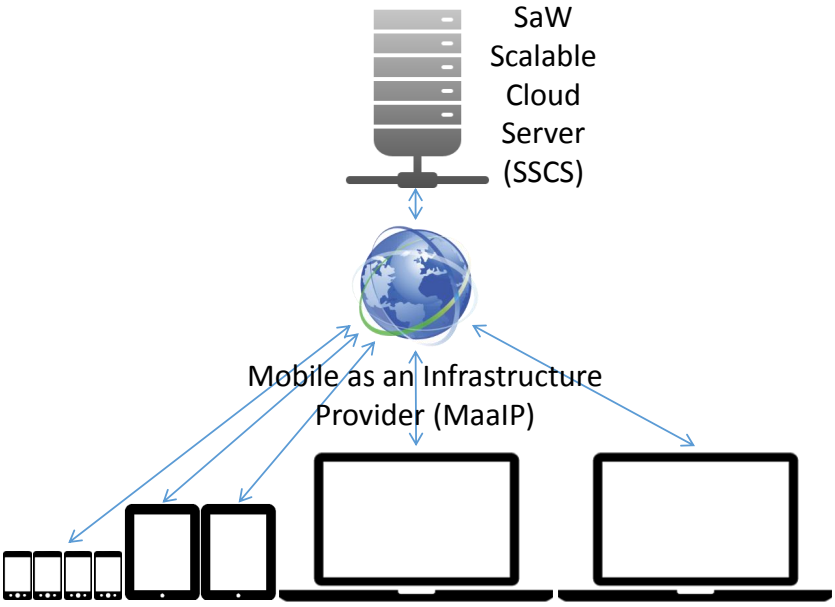


**Figure 3.12:** General SaW system architecture diagram

Server (SSCS) which manages server resources in order to provide a consistent, scalable and a single service front-end to the clients. It deals with balancing the load through the different available servers. The SaW client-side is completely Web browser oriented. Hence, emerging technologies such as HTML5, JavaScript, WebGL or WebCL play a crucial role by providing interoperability to cope with hardware and software heterogeneity.

All the computing and data transmission overhead in the client-side cannot affect the experience of the consumed content. Hence, on a first step, the SaW system has to create a device capabilities profile. To this end, the server adds in the first response to the client a benchmarking test in order to assess the processing capabilities and the hardware assets disposal of the user device. The score is sent to the task distribution manager of the SSCS, which decides the complexity of the background image analysis tasks that fit into that client following the global task distribution strategies.

On a second step, once the SSCS has set specific tasks to be run on a suitable client device, a data transfer is initiated from the server with the image frame and the image processing JavaScript script or scripts. These are classified by complexity and invoke different technologies such as WebGL or WebCL to exploit the GPU and/or multi-core assets of the device. The client applies the scripts over the images as a background process, avoiding any user experience damage. The computed results are sent back to the SSCS and it harvests, formats and stores all the

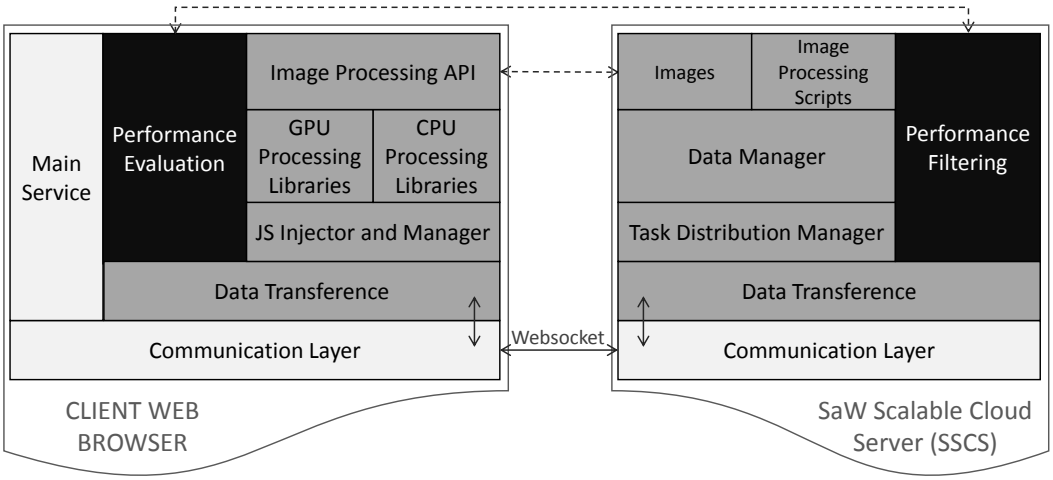


**Figure 3.13:** SaW Client-server block diagram and its communication

incoming image computing outcome to be mined later by the service provider. While a user is enjoying a social service similar to YouTube or Vimeo, SaW allows the service to deliver independent image analysis tasks queued to the different clients until each session finishes. In case the server does not receive a result in an elapsed maximum time from a specific client, it considers that device is not available anymore and queues it to another one. Figure 3.13 depicts a more detailed client-server SaW architecture and the communication between them.

**Client Web Browser SaW Architecture**

The SaW approach is designed to run the client-side application over a standard Web browser composed by the following modules (see Figure 3.13):

*Main Service*: This is the main application of the service provider and gates what the user wants to consume. Note that a client using the Main Service has committed to join SaW by allowing service providers to gain idle resources in the user's device to add background activities while preserving a good Quality of Experience (QoE).

*Communication Layer*: This module enables the communication between the client and the server with widely supported Web communication protocols: Websocket and AJAX. The WebSocket Protocol enables two-way communication between a client and the server. Here the security model used is origin-based that is widely used by Web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based services that need two-way communication with servers that does not rely on opening multiple HTTP connections

[Fet11]. Even if the implementation of the WebSocket protocol is widely implemented and on the roadmap of all the Web browsers, nowadays there are some restrictions to use Websockets over some devices, such as mobiles. Anyway, a less efficient polling approach with Ajax is a feasible alternative to WebSocket. Ajax is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, Web applications can send data to, and retrieve data from, a server asynchronously keeping visual fluidity and behavior of the foreground Web application [Gar05].

*Performance Evaluation*: This module launches a performance test at the beginning of the application runtime in order to profile the capabilities of the device in terms of CPU, GPU and other features such as the autonomy of the battery or the bandwidth. Then, the benchmark results are sent to the SSCS and it settles the complexity threshold for image processing that this device can deal with considering all the discovered aspects. This evaluation test is not repeated during an active session but it could be performed again to tune the background tasks to a new context, specially if the main service is very changeable from a processing requirement perspective.

*Data Transference*: This module works hand in hand with the Communication Layer, dealing with the data transference between the client and the SSCS. HTML5 Web Storage facilities are used to create and maintain the incoming data. On the one hand, the client receives a new image for each image processing task and one or various scripts to be applied for that image. These are stored locally and once the processing is over, this module fits the format of the results to transfer them to the SSCS. It is important to highlight that SaW runs entirely in the memory of a Web browser.

*JS Injector and Manager*: It takes charge of handling the scripts received from the server. This module injects and deletes the scripts on runtime without interfering on the user experience and prepares them to execute the background tasks in an optimal way. It is very close to the following two modules enabling the scripts to take advantage of the GPU and CPU resources of an appliance.

*CPU Processing libraries*: Depending the performance ranking, the server decides the scripts delivered to the client oriented to exploit CPU resources. This module manages the CPU processing JavaScript libraries enabling multi-core tasks through WebCL.

*GPU Processing libraries*: This module wraps the JavaScript libraries ready to foster hardware acceleration by the GPU of the client device using WebGL

and WebCL technologies. The availability of these assets obeys the performance assessment that are essential due to the high efficiency of the GPU oriented computing.

*Image Processing API*: This core layer provides the Web application an API to perform the image processing scripts on the client side. It runs image analysis tasks in the background on top of the JS Injector and Manager and using the CPU and GPU processing libraries.

**SaW Scalable Cloud Server Architecture**

The SSCS has different modules to manage all the SaW service infrastructure on the server side. These elements are presented on Figure 3.13 and briefly explained below:

*Communication Layer*: It manages the communication between the SSCS and the client. With the same functionality mentioned in the client side, this module is deployed on top of WebSocket and AJAX protocols.

*Data Transference*: It is supported by the Communication Layer and it is the responsible for exchanging the data with the client (e.g. the images and the scripts for each processing task).

*Task Distribution Manager*: This block has the global view of the SSCS to categorise and dispatch all the image analysis tasks that the service provider wants to perform through the clien device community. It splits and queues the processing jobs by connecting an image with some specific scripts and estimating the complexity of each computing work. It collaborates with the Performance Filtering module and requests and exchanges data with the Data Manager module.

*Performance Filtering*: This element receives the performance assessment from the clients and analyses the capabilities of the devices to inform the Task Distribution Manager module, who assigns a specific task to that device gaining specific hardware (GPU or CPU) acceleration according to its assets disposal.

*Data Manager*: This block manages all the data involved in the SSCS interrelated from different data-bases containing the images to be processed, the image processing scripts (some of them to be run over CPU architectures and others over GPU ones) and the results obtained by the clients. At the end, it links an outcome to data (one image) and logic (one script).

#### 3.4.3.4 Considerations regarding the Design Objectives

**Elasticity**

Saw assures elasticity through Performance Evaluation and Performance Filtering modules. They profile the capabilities of the client devices in terms of CPU and GPU, but also regarding other features such as the level of battery, which is one of the most critical parameters on mobile devices. For this reason, SaW will not only consider the processing capabilities of the client devices, but also other features in order to provide flexibility to the service and let the service provider to parametrise the QoE by terms of elasticity. The elasticity parameter is considered in Section 3.4.4.2.

**Performance**

Due to the relevance of the performance in the SaW system, Section 3.4.4.2 presents a performance modeling with a validation that demonstrates the feasibility of the SaW approach, and the cost-saving of complementing a cloud service with a mobile grid as an infrastructure.

**Security and Privacy**

SaW takes into consideration the security aspects regarding confidentiality and integrity. Those are assured by the well-known standard mechanisms of authentication, authorisation, encryption and auditory. As mentioned above, in SaW a client has to commit the hiring of its device resources in order to access the social media service. Thus, reciprocity conditions concerning privacy and security should be observed by the registered client and the server.

It is mandatory to verify social identity of the computation node to check its rights and permissions. The use of a centralised mechanism eases handling frequent user access privilege updates (such as invitation or revocation of access rights) in large dynamic systems like social networks.

Once the trustworthy handshake has been done, the data must be encrypted to prevent man-in-the-middle attacks. SaW deploys a temporal token based solution to limit access permissions and encrypts the data flows, with TLS protocols, for the Web communication layer.

Concerning the security threats, a push design lets SaW to prevent search over the database and DoS attacks. Note that in a pull model, an attacker, a malicious computation node, could get a promiscuous mode by notifying a permanent idle status to retrieve a set of processing tasks and associated data (crawling for later search) or to capture all the queued tasks (turning the uniformly distributed dispatching management into a burst one for a DoS attack). To prevent this behavior, in the push model of SaW, the cloud broker employs the queue of batch jobs to delegate them. This way, it is difficult for the nodes to search a specific data or

claim a particular task. Moreover, server authentication could be addressed using a Synchronised Token Pattern (CSRF Token) that prevents against Cross-Site Request Forgery (CSFR) attacks [BJM08].

Concerning possible privacy policies, privacy levels and accuracy can be defined differently within a social setting. This avoids the Task Distribution Manager to send private media content to a non-allowed end-user, even to process it in the background. For this purpose, SaW considers three types of media scopes with different set up implications: public, widening the media analysis to any available device; shared with friends, limiting the trusted area to the devices inside the social acquaintance circle; private sharing, constrained to a specific list of computing nodes from the cloud to manipulate data.

### 3.4.4 Evaluation

In this Section a proof-of-concept implementation of SaW architecture is described, providing experimental evaluation results and an analysis of the performance based on a previous model. The evaluation is focused in two different aspects:

- The scalability of the SaW approach, with a specific comparison between the involved Web technologies: WebGL and WebCL.
- The performance behaviour of the system when considering different types of client devices, according to the target SaW scenarios. This includes a model of the computational cost that considers CPU, GPU and communication resources, as well as some performance figures obtained for different scenarios with realistic combination of device types.

#### 3.4.4.1 WebGL and WebCL scalability comparison

This subsection presents the experimental results of using WebGL and WebCL technologies in order to explore the scalability of current Web browsers to exploit GPU resources. A proof-of-concept implementation of the SaW testbed has been developed to distribute a queue of 100 tasks over a different number of clients with identical capabilities. Using homogeneous devices enables to measure the scalability without loss of generality. The heterogeneity of the devices is addressed in next subsection.

For that purpose, a SSCS implementation has been built with a combination of Node.js and MongoDB to obtain a low latency server and to be able to deal

with high concurrency requests. Both technologies provide event-driven systems that enables a non-blocking I/O model that makes it lightweight and efficient in high concurrency environments with a NoSQL data structure. Three different instances of the server have been deployed in order to avoid bottlenecks and provide sufficient resources for the different clients.

In the client side, according to the proposed architectural design, our implementation works over Web standards to cover a wide set of devices and follows a modular design. These modules enable a real-time communication with the server and are able to inject JavaScript libraries in runtime for the background tasks.

As clients, we used a different number of identical PCs with the following capabilities: Windows 8.1 Intel(R) Core(TM) i5-3330 CPU @ 3.00GHz with Intel(R) HD Graphics 2500. To test WebGL, Firefox 42.0 Web browser has been used, that enables WebGL by default [can]. Currently, there is no native support for WebCL in Web browsers. Thus, an experimental extension [Nok] has been used on top of a portable Firefox 22.0 Web browser.

The server creates a queue of 100 tasks with an image and an associated algorithm for each image. The sever dispatches the tasks to the available clients. Since all the clients have the same capabilities, the tasks are homogeneously distributed through all of them.

The performed algorithm computes the DITEC method (Trace transform based method for color image domain identification) [OQFS14] by means of algebraic operations such as matrix dot products that can be highly parallelised at different states (per frame, per angle during the Radon Transform operation, etc.).

We have evaluated two different SaW implementations with the aforementioned workload, the first one using WebGL and the second one using WebCL. In the performed experiments the workload has been distributed among a number of workers going from 1 to 20. The same queue of tasks has been also performed on a single PC with the same capabilities using OpenGL and OpenCL instead of doing it from the Web browser. The results obtained are shown in Table 3.4.

Comparing the values described in Table 3.4, obtained over the same PCs, of using a local server with OpenGL and OpenCL, with a single worker in the distributed approach with WebGL and WebCL respectively, it can be said that the local approach obtains better results. The reasons are mainly two: (1) the latency introduced by the delivery time of the image, the script and the results between

the SSCS and the client in the distributed approach, and (2) the performance gap of the bindings of WebGL and WebCL to exploit the hardware resources in comparison with OpenGL and OpenCL.

From the obtained values in Table 3.4 regarding the distributed approach with different number of clients, it can be inferred that (1) the speedup is very high for both implementations, which denotes that the parallelisable fraction of the workload is very big, as expected for the described SaW use case; (2) WebCL implementation performs better than WebGL, and (3) the speedup obtained for the WebCL version is not as high as the one obtained for WebGL.

**Table 3.4:** Computational cost in terms of time for the same workload for a local sever with OpenGL and OpenCL, and for a number of workers from 1 to 20 in the distributed approach with WebGL and WebCL

**LOCAL SERVER**

| **Number of workers** | **Computational time in $ms$ with OpenGL** | **Computational time in $ms$ with OpenCL** |
|---|---|---|
| 1 | 132,570 | 61,780 |

**DISTRIBUTED APPROACH**

| **Number of workers** | **Computational time in $ms$ with WebGL** | **Computational time in $ms$ with WebCL** |
|---|---|---|
| 1 | 275,295 | 111,300 |
| 2 | 161,820 | 73,765 |
| 5 | 63,260 | 34,290 |
| 10 | 34,476 | 19,740 |
| 20 | 17,825 | 11,120 |

Using Amdahl's Law [Amd67] we have calculated the parallelisable fractions of the workload for both WebGL and WebCL versions, which have resulted 98.87% and 94.36% respectively. To obtain these values, we have excluded the measures obtained with one single worker, since it does not reflect the task scheduling effects of managing different workers and consolidating the results. As shown in Figure 3.14, Amdahl's Law interpolates real measures with reasonable fidelity for the range [2, 20] of workers. Accordingly, we have used this approximations to predict results for 50 and 100 workers (see Figure 3.14). As appreciated in the Figure, WebCL would outperform WebGL until near 100 workers.

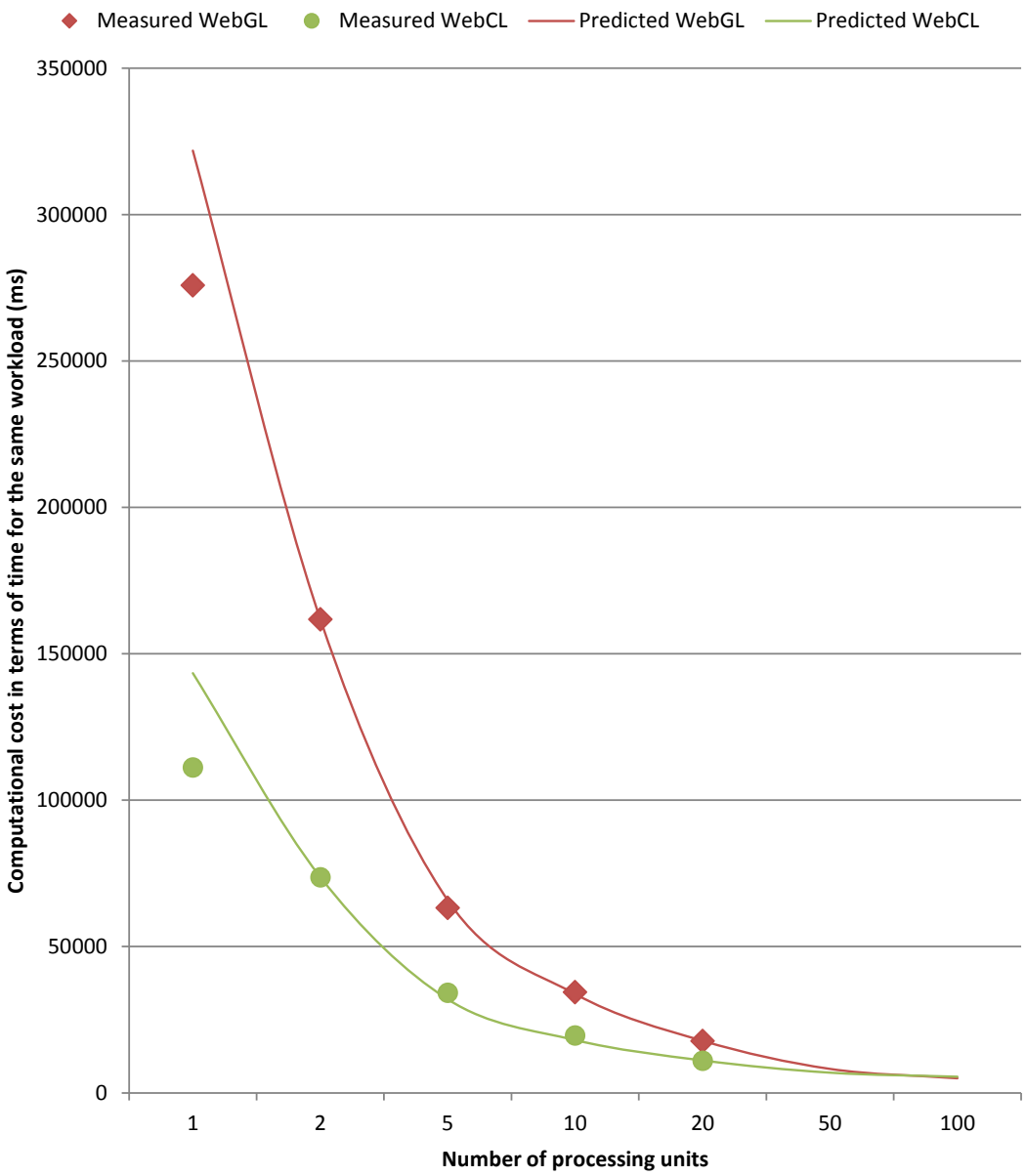


**Figure 3.14:** Computational cost in terms of time for a different number of workers in terms of processing units in the distributed approach for WebGL and WebCL. The real measured values, presented in table 3.4, are shown for a range of workers from 1 to 20, while predicted values, following Amdahl's Law, are shown for a range of workers from 1 to 100.

#### 3.4.4.2 Performance Modeling with heterogeneous devices

In this subsection we present a performance evaluation model of the SaW approach based on the model published in [ZMT+13] taking into account different type of devices. More specifically, here we take into consideration both CPU and GPU resources, while in [ZMT+13] only CPU resources were contemplated.

The performance of SaW can be analysed by following the Bulk Synchronous Parallel (BSP) model [Val90]. The BSP model is a generalisation of the classical PRAM model [FW78] for shared memory.

As presented in [ZMT+13], we will consider 3 different kind of devices as processing units for the distributed approach: smartphones, tablets and PCs. Table 3.5 shows different connectivity and processing power values for each one of the device types based on market surveys [Tom11] [Leg12]. The table includes information about a server in order to give a comparative estimation of the computational cost. According to market surveys [str], 50% of the PCs and 30% of tablets have a GPU available, while it decreases until 10% in the case of the smartphones.

**Table 3.5:** Estimated processing and communication properties for different type of devices

| ID | Device | Connect. | Bandwidth $\hat{b}_i$ | CPU GFlops $\hat{F}_{ci}$ | GPU GFlops $\hat{F}_{gi}$ |
|---|---|---|---|---|---|
| (m) | Mobile phone | UMTS | 3Mbit/s | 0.05 | 0.2 |
| (t) | Tablet | Wifi | 8Mbit/s | 0.08 | 0.32 |
| (p) | PC | DSL | 20Mbit/s | 2.5 | 10 |
| (s) | Server | SATA | 6Gbit/s | $p_s$x82.8 | – |

Equation 3.9 extends the Equation 6 of [ZMT+13], which considers the total computational cost ($C_T$) as the time to perform a workload unit distributing it across all the different type of devices in parallel.

$$C_T = \left( \sum_{i=1}^{n} \frac{1}{C_{ci}} + \sum_{i=1}^{n} \frac{1}{C_{gi}} \right)^{-1} \tag{3.9}$$

The equation divides the partial computation time cost for each type of device to perform their part of the workload ($C_i$) in two:

- $C_{ci}$: the partial computation time cost for each type of device that only have CPU processing capabilities to perform their part of workload.
- $C_{gi}$: the partial computation time cost for each type of device that have GPU and CPU processing capabilities to perform their part of the workload.
- $n$: the number of different type of devices. Note that according to the information of Table 3.5, $n$ will be 3 since the table defines three type of devices for the distributed approach: smartphones, tablets and PCs.

Equations 3.10 and 3.11 represent $C_{ci}$ and $C_{gi}$ respectively, extending the equation 7 of [ZMT+13] and assuming sufficient resources in the server side,

$$C_{ci} = \frac{W_i}{f_{pci} \cdot \hat{F}_{ci} \cdot p_{ci}} + \frac{g_i}{f_{bi} \cdot \hat{b}_i \cdot p_{ti}} + \hat{m} \cdot p_{ci} \tag{3.10}$$

$$C_{gi} = \frac{W_i}{f_{pgi} \cdot \hat{F}_{gi} \cdot p_{gi} + f_{pci} \cdot \hat{F}_{ci} \cdot p_{ci}} + \frac{g_i}{f_{bi} \cdot \hat{b}_i \cdot p_{ti}} + \hat{m} \cdot p_{gi} \tag{3.11}$$

where:

- $W_i$ is the computational workload in terms of the computation time assigned for device type $i$, to be distributed among all the different available processing units of device type $i$.
- $g_i$ is the communication workload in terms of number of bytes of information to be transmitted from the server to the devices of type $i$.
- $\hat{m}$ is the estimated cost in terms of computation time to establish a new task to a processing unit and its management.
- $\hat{b}_i$ is the estimated bandwidth for device type $i$ (see Table 3.5).
- $\hat{F}_{ci}$ is the estimated CPU processing capability in terms of flops for device type $i$ (see Table 3.5).
- $\hat{F}_{gi}$ is the estimated GPU processing capability in terms of flops for device type $i$ (see Table 3.5).
- $p_{ci}$ is the number of different processing units of device type $i$ with only CPU capability.

- $p_{gi}$ is the number of different processing units of device type $i$ with both GPU and CPU capabilities. Note that following the assumption that all the devices with GPU capability will also have CPU capabilities, in Equation 3.11 $p_{ci}$ and $p_{gi}$ will be the same number of processing units.
- $p_{ti}$ is the number of messages exchanged between the processing units of type $i$ and the server.
- $f_{pci}$ is an elasticity factor introduced to determine the percentage of the CPU to be used from the available CPU resources of device type $i$.
- $f_{pgi}$ is an elasticity factor introduced to determine the percentage of the GPU to be used from the available GPU resources of device type $i$.
- $f_{bi}$ is an elasticity factor introduced to determine the percentage of the bandwidth to be used from the available bandwidth for device type $i$.

As presented in [ZMT+13], the same model can be used for a multi-core server approach in order to give a comparative estimation with the distributed approach. Equation 3.12 shows the cost of a multi-core server in terms of time ($C_s$) with $p_s$ processing units sharing a single memory:

$$C_s = \frac{W}{f_{ps} \cdot \hat{F}_s \cdot p_s} + \frac{g}{f_{bs} \cdot \frac{\hat{b}_s}{p_s}} + \hat{m} \cdot p_s \tag{3.12}$$

In order to compare a distributed computing approach with a dedicated local server, different $C_T$ and $C_s$ have been calculated applying the aforementioned model. In order to avoid any annoyance in the user experience, the elasticity factor has been set to 0.15 both for bandwidth ($f_b$) and for processing power ($f_p$) for the background activities of the distributed approach. However, we considered that the local server is exclusively dedicated to these tasks ($f_{ps}$=$f_{bs}$=1).

Figure 3.15 shows four different cases of performance behavior for several values of model parameters. A lineal increment of processing units is compared for different values of $W$, $g$ and $\hat{m}$.

As it can be observed in figure 3.15a, while the total distributed cost decreases for more devices, the local server starts to increase after reaching a minimum with 30 workers. This reflects that the data communication bus on the server is a bottleneck while, in the distributed solution, the servers should have enough bandwidth to provide the required bandwidth for each device. This trend is more clear in figure 3.15b, with an increased communication cost ($g$).

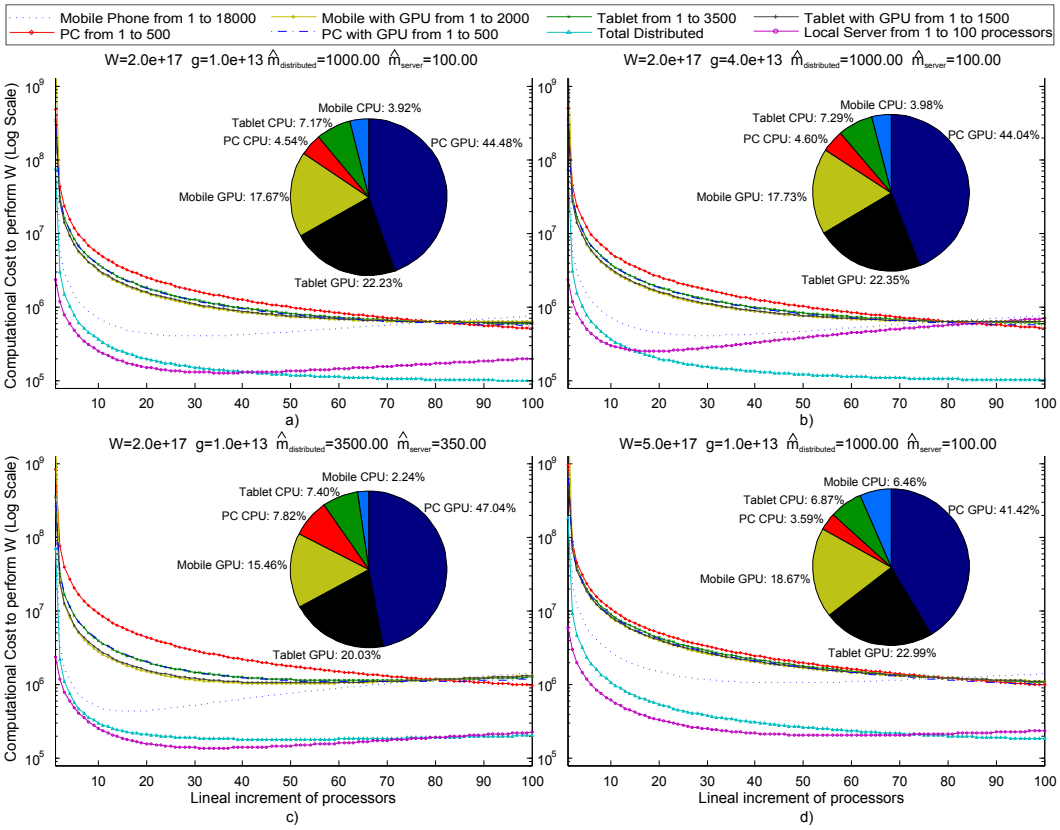


**Figure 3.15:** Computational cost estimation under different sizes of work load ($W$) communication cost ($g$) and task management cost ($\hat{m}$: $\hat{m}_{distributed}$ for the client devices and $\hat{m}_{server}$ for the server). The pie chart presents the load balance between the different devices in the distributed approach to have the same computational cost at 80% of the X axis.

The efficient management of all the created tasks becomes a critical factor as well. This is reflected in Figure 3.15c, where $\hat{m}$ has been increased comparing to Figure 3.15a. The distributed approach has to manage thousands of devices while the local server goes from 1 to 100 workers so the management cost is more damaging to the SaW-based distributed solution.

If we increase the global workload ($W$), maintaining the same communication cost, a bigger gap between the workers in the local server and the devices in the distributed approach is needed to push the distributed solution performance ahead. This is reflected in Figure 3.15d comparing to Figure 3.15a. To sum up, with a enough size of user's devices partially dedicated to social service improvement it is possible for a SaW system to lead traditional server based performance. Moreover, the elasticity factor that has been set in a conservative way can be increased considerably in many scenarios without damaging the user experience.

#### 3.4.4.3 **Remarks**

In this section some remarks are reflected regarding the validation of the SaW hypothesis that has been introduced in Section 3.4.1. Recall first that the SaW approach is oriented to complement a cloud server, and not intended to beat it in computational performance. In this regard, the delay-tolerant nature of the tasks to be distributed to the clients, such as in a video tagging scenario, plays in favour of the SaW approach.

As an example, consider a task that will require a time $t$ to be executed in the server, and, upon the results obtained in Section 3.4.4.1, assume for the workload a parallelisable fraction of about 99%. This means that the server has to spend about 1% of $t$ for the distribution overhead, represented for the third term in Equations 3.10 and 3.11. In other words, the server would be able to manage the distribution of about 100 of these tasks in the computational time of $t$.

Continuing with the same example, assume now that the server is dedicated exclusively to task distribution, and that the SaW ecosystem is composed only of smartphones with about 1/50 of the processing power of the server according to Table 3.5. Note also that to preserve QoE, the smartphones will work with an elasticity factor, say it 0.15 to be conservative, which leads to a processing capability for each one of the smartphones of 0.003 of the server power. For a set of 100 smartphones, the server will take a processing time of $t$ to distribute the queue of 100 tasks, and will obtain all the results back in a time lapse of hundreds of $t$ from the smartphones. This concludes that the server consumes only resources for

time $t$, equivalent to perform a single task, and will asynchronously obtain the results for 100 tasks instead. However, the time period will be of hundreds of $t$.

Nevertheless, note that the example above reflects a worst-case scenario, according to the current use cases and user habits already mentioned. In a more realistic scenario, like the ones presented in the former subsection, the SaW approach would elastically distribute the workload among the different devices according their features (such as smartphones, tablets and PCs with CPU and/or GPU processing capabilities).

To summarise, the exchange of "time-for-resources" or, from the service provider perspective, "time-for-money" explained through the above example is in the core of the SaW approach, since delay-tolerance and elasticity provides sufficient freedom degrees in usual scenarios. Finally, the technological evolution also plays in favour of the approach. The performance gap between the different type of devices has been continuously reduced in the past and still continues. Besides, improvements in the bindings of Web browsers to support WebGL and WebCL can also be expected, which will result in a more efficient use of the hardware resources.

### 3.4.5 **Conclusions**

In this paper we have introduced the concept of Social at Work, SaW, which aims to complement a Web-based social media service with all the idle devices, mostly mobiles, that usually have underexploited resources while accessing the service. SaW proposes a Mobile as an Infrastructure Provider (MaaIP) model, creating a system related to Mobile Grid Computing concept with the available CPU and GPU resources of the different client devices, to complement a virtualised cloud server providing the social media service.

Aimed to achieve enhanced and automatic media tagging over social media datasets, SaW fosters background dispatching of media analysis over connected clients, providing a high elasticity and dealing with the availability of the resources related to the spontaneous presence of users. Then, SaW copes with hardware-accelerated image processing tasks execution in background, according to the capabilities of each device. The computing tasks are embedded in the foreground social content without draining the users' bandwidth or affecting to the perceived Quality of Experience. In harmony with the presented scenario, delay-tolerant background tasks enable the SaW approach to exchange "time-for-resources" or "time-for-money". This means that mobile devices, instead of being as resource

intensive as servers, can dedicate the sufficient time to perform the task, preserving the QoE according to their capabilities, and saving cloud costs to service providers.

SaW deploys a powerful pure Web platform for video analysis by means of exploiting high user availability density, and the explained capability to run scripts in background threads of Web browsers. Therefore, the SaW concept targets a device community as a processing grid removing the need for install client applications, adding a delivering computing layer to the stack of the HTML5-based main service instead.

In order to evaluate the approach, we have developed a proof-of-concept implementation of SaW, including versions for existing WebGL and WebCL technologies. Results of the experiments show the high speedup obtained by parallelisation, which confirm the scalability of the approach exploiting GPU resources from Web browsers with both WebGL and WebCL technologies. The scheduling elasticity in the server side has been designed to take advantage from the delay-tolerant target scenarios, with a heterogeneous community of client devices characterised by the assorted availability of resources.

This paper has extended a previous performance model that was focused only in CPU resources, to consider both CPU and GPU capabilities. The model allows to predict the performance of a distributed system including diverse client devices, which have been illustrated through a set of example configurations, in comparison with a local server solution. The maximum benefit is obtained for higher delay-tolerant computational load, with independent tasks able to be distributed to idle devices, being able to compensate the task scheduling management and consolidation overload of the server. The technological evolution, with a clear trend to reduce the performance gap between laptops and mobile devices, as well as to improve the efficient exploitation of hardware resources from a Web browser, favours the SaW approach.

As a summary, SaW deploys a social distributed computing infrastructure on top of pure Web-based technologies, building a grid of resources to perform background media analysis tasks leveraging hardware-acceleration for a social media service.

# Part III

# Conclusions

CHAPTER

# 4

# Conclusions

This research proposes interoperable technologies for multi-device media services. On the one hand, three main challenges have been addressed to enable creation and delivery of multi-device media services: **multi-device adaptation**, **cross device synchronisation** and **multi-connection**.

Regarding **multi-device adaptation**, a Web-based distributed adaptation architecture for multi-device applications driven by media content has been proposed as a solution. This interoperable and standard approach allows broadcasters and media application developers to easily create this kind of applications by extending the Web Components standard. It also provides broadcast-Internet convergence through Web technologies, following the trend of the broadcast-related standards for the delivery of hybrid services, such as HbbTV. Moreover, the proposed approach improves the process that broadcasters and application developers currently need to implement, distribute and maintain over a set of complex technical solutions tailored to each specific target platform. The solution enables the following aspects:

- A distributed architecture that allows each client to make autonomous decisions according to a shared context and still provide a consistent view to the user through multiple displays simultaneously.

- A single development environment for broadcasters and application developers that facilitates the creation of multi-device applications without dealing directly with all the adaptation challenges and possible context

situations, enabled by the expressiveness of the approach on top of Web technologies.

- The design of generic or custom adaptation rules to create multi-device responsive applications, boosting the re-usability and the development of this kind of applications, due to the openness of the solution built on top of standard technologies.

A proof-of-concept implementation of the architecture has been also presented, including some adaptation testing rules to validate the design objectives of the architecture and its modularity and flexibility. These experiments demonstrate distributed decision making with autonomously executed adaptation engines, tracking dynamic setups automatically and performing real-time self-organising. The experiments also validate that the proposed scenarios can be achieved by creating adaptation rules on top of our architecture. The results reflect the openness of the proposed approach on top of Web Components and the expressiveness of the properties added for adaptation purposes built over standard technology.

In terms of **cross device synchronisation**, this research proposes a cloud session maintenance architecture as a solution to share context and timing information to synchronise multi-device media services, such as an HbbTV television with mobile devices.

Instead of turning on local network communication protocols to pair the second screen devices with the TV set, a server-based architecture has been proposed that pushes to the cloud all the shared information for synchronisation. It removes local area network constraints. Thus, devices could be in different networks and still provide a synchronised experience.

A proof-of-concept implementation has been provided, as well as validation experiments of the proposed architecture. The validation has been done in terms of latency and to parameterise the server performance for a high volume of users accessing concurrently. The results conclude that the latency of the proposed architecture is suitable for most of the second screen services that can be envisaged over this solution. On the other hand, when incrementing the number of simultaneous connections that are exchanging information with the server, there is a stable zone where the latency value remains quality threshold. This shows that the server architecture is ready to perform a multi-device ecosystem of many

users interacting with synchronised contents concurrently. Thus, in order to manage adequately to a volume of concurrent requests for the latency stable zone, the server just must dimension CPU available hardware.

In essence, a suitable end to end solution based on a cloud session maintenance architecture has been provided to enable cross device synchronisation for multi-device media services. This architecture performs a solution over already commercialised devices such as HbbTV compatible Connected TVs (v. 1.1 or v.1.5) and Android, iOS, Windows, Blackberry, etc. smartphones and tablets with a standard mobile Web browser.

Regarding **multi-connection**, different mechanisms have been proposed as a solution to discover and associate devices for multi-device media services, such reaching a second screen around an HbbTV television. Three different discovery and association mechanisms have been presented landed by the implementation of spatial discovery scope approaches regarding the physical co-location.

On the one hand, visual solutions are proposed to deliver the association parameters, represented by widespread QR code technique. It requires a second screen device with a camera and a native application for the QR code scanning. Some experiments are presented to evaluate the main factors involved in the QR code scanning process from a mobile device to a TV.

On the other hand, sound patters are also proposed as a mechanism to deliver the association information from one device to another seamlessly. In the case of HbbTV televisions, due to the necessity to generate a unique ID for each TV, the sound pattern should be generated in the TV application itself and the paper analyses the current problems to overlap an HTML-based audio to the broadcasted signal in HbbTV. Experiments are presented to measure the interruption time of the mainstream media since the broadcast signal goes to background, until it is back to foreground after played a sound pattern recognisable by the second screen device.

As the third mechanism, a bumped-based solution is presented in the paper to discover and associate multiple devices. It is based on a synchronous event-driven server that receives the different requests and associates them depending on the reception timestamp and some additional information (the location of the devices and a button-hopping mechanism). The research presents some experiments measuring the delay between the different requests received by the server through various networks.

Finally, a three step solution based on the combination of the previous mechanisms is presented as the most user friendly and robust. All the experiments confirm the feasibility of the approach as a suitable solution for the discovery and association of multiple devices, such as a mobile device around an HbbTV television in broadcast-related second screen services.

Apart from the three main challenges addressed to provide interoperable technologies for multi-device media services, a solution to provide a two-way complementarity of computing capabilities has been proposed. This solution meets the computing requirements that media services demand in both directions: providing the **3DMaaS solution** as a hybrid local-remote rendering approach to enhance the real-time computing requirements of client devices with cloud resources, and providing the **SaW solution** to complement a cloud infrastructure for delay-tolerant computing requirements with a set of connected client devices, saving costs to the service provided at the server side.

The proposed **3DMaaS solution** deploys remote servers performing the remote rendering of complex 3D scenes and then sending the frame results to the client device. This video streaming server approach pushes part of the graphics generation logic to the cloud and, in essence, enables complex media rendering in thin devices. Driven by latency constraints, the solution proposes a hybrid approach combining remote rendering of background 3D objects, where the latency does not have a high impact on the user experience, with local browser WebGL rendering of foreground 3D objects which require low latency. Synchronization and 3D scene consistence challenges must be managed by the HTML5 application, and cross device synchronisation mechanisms could be used for this purpose. Experiments show the results obtained by the proposed system for hybrid remote and local rendering enhance the interactive experience of 3D graphics on digital home devices proving the feasibility of interactive navigation of high complexity 3D scenes while provides an interoperable solution that can be deployed over the wide device landscape.

Finally, this research has also introduced the concept of Social at Work, called **SaW solution**, which aims to complement a Web-based social media service with all the idle devices, mostly mobiles, that usually have underexploited resources while accessing the service. SaW proposes a Mobile as an Infrastructure Provider (MaaIP) model, creating a system related to Mobile Grid Computing concept with the available CPU and GPU resources of the different client devices, to complement a virtualised cloud server providing the social media service.

Aimed to achieve enhanced and automatic media tagging over social media datasets, SaW fosters background dispatching of media analysis over connected clients, providing a high elasticity and dealing with the availability of the resources related to the spontaneous presence of users. Then, SaW copes with hardware-accelerated image processing tasks execution in background, according to the capabilities of each device. The computing tasks are embedded in the foreground social content without draining the bandwidth of the devices of the users or affecting to the perceived Quality of Experience. In harmony with the presented scenario, delay-tolerant background tasks enable the SaW approach to exchange time-for-resources or time-for-money. This means that mobile devices, instead of being as resource intensive as servers, can dedicate the sufficient time to perform the task, preserving the QoE according to their capabilities, and saving cloud costs to service providers.

SaW deploys a powerful pure Web platform for video analysis by means of exploiting high user availability density, and the explained capability to run scripts in background threads of Web browsers. Therefore, the SaW concept targets a device community as a processing grid removing the need for install client applications, adding a delivering computing layer to the stack of the HTML5-based main service instead.

In order to evaluate the approach, a proof-of-concept implementation of SaW has been implemented, including versions for existing WebGL and WebCL technologies. Results of the experiments show the high speedup obtained by parallelisation, which confirm the scalability of the approach exploiting GPU resources from Web browsers with both WebGL and WebCL technologies. The scheduling elasticity in the server side has been designed to take advantage from the delay-tolerant target scenarios, with a heterogeneous community of client devices characterised by the assorted availability of resources.

The SaW solution has extended a previous performance model that was focused only in CPU resources, to consider both CPU and GPU capabilities. The model allows to predict the performance of a distributed system including diverse client devices, which have been illustrated through a set of example configurations, in comparison with a local server solution. The maximum benefit is obtained for higher delay-tolerant computational load, with independent tasks able to be distributed to idle devices, being able to compensate the task scheduling management and consolidation overload of the server. The technological evolution, with a clear trend to reduce the performance gap between laptops

and mobile devices, as well as to improve the efficient exploitation of hardware resources from a Web browser, favours the SaW approach.

As a summary, this research work provides progress beyond the state-of-the-art for interoperable technologies for multi-device media services, presenting five main contributions on different aspects. Three of them propose solutions to create a single application code that will spread to multiple devices, offering the user a consistent experience through multiple devices at the same time. For that purpose, those contributions address the multi-device adaptation challenge, a cloud-based state sharing for synchronisation, and mechanisms to discover and associate devices. The other two contributions pose solutions to meet the computing requirements of the media content itself. One of them provides a solution to enhance the rendering capabilities of thin devices in real-time by terms of a hybrid local-remote system. Finally, the last contribution presents a solution to exploit the computing capabilities of the idle client devices connected to a social media service, saving delay-tolerant computational workload in the server infrastructure.

## 4.1 Future Work

The research can be extended in several directions. On the one hand, an ongoing topic has been started during the dissertation regarding adaptive user interfaces. As discussed previously, multi-screen systems enable broadcasters and application developers to deliver their users more content related to the mainstream media. They also provide opportunities for controlling the experience as well as to offer the user new ways for personalisation, social sharing and consumption of media content from various sources [CBJ08]. Regarding multi-device adaptation, in Section 2.2 a Web-based distributed adaptation architecture for media-driven multi-device applications has been proposed, enabling broadcasters to provide the users a single experience through multiple devices at the same time. However, besides obvious advantages delivered by multi-screen media, more screens demand higher cognitive load for viewers to understand what they watch and to correlate content from different video streams. The required visual attention also increases and needs to be distributed across multiple displays situated at various locations in a three dimensional space [VM15].

In this context, the user interface becomes a key factor in order to provide the users a good experience across multiple devices at the same time, facilitating

the understanding of the application and providing an intuitive interaction way. When broadcasters or application developers are facing a single-device user interface, they typically define a template to organise the items in the layout, usually describing a different behaviour of the layout for each target device. For instance, in Web applications developers can provide a CSS template with Media Queries that will adapt the user interface of the application to the final features of the device.

However, when developers are dealing with multi-device applications, this approach becomes unachievable as stated in Sections 2.2 and 2.3. In a multi-screen solution like the proposed one, the system will dynamically decide which elements of the application will be shown on each device, splitting the application through different devices simultaneously, in order to provide a consistent and coherent view to the user. This means that the list of elements available on each device will depend on the changeable context of the users and will update dynamically when a new device is added or removed. In this scenario, it will be tedious, very expensive and unaffordable for developers to provide an explicit template to organise the elements on a user interface for all the possible combinations and moreover for each target device. The proposed multi-device adaptation solution provides arbitrary divisions to arrange the elements in a responsive user interface following some given hints and rules, creating a specific layout template depending the circumstances.

However, it is not easy yet for broadcasters and application developers to create a set of rules and hints to select the best user interface layout depending some contextual information, since they are not used to this contextual-based rules. Nevertheless, there are aspects in interface design, such as the functionality and usability that are well known aspects in the field of Human Computer Interaction (HCI), where the developers have a stronger criteria [MK06]. For example, for a background device such as a TV, developers might want to allow overlaps between the different elements with Picture-in-Picture-like interfaces, while would avoid to have elements below the visible area of the screen to avoid scrolling. This pattern could be completely different for a touchable device, in the same way that it will be very dependant on the nature of the media application and its elements. Moreover, in addition to the traditional HCI parameters, there is a new wave in the field emphasizing the importance of aesthetic aspects in interface design for the users' likability and system acceptability [BL08] [AL11].

In this context, the discussion of the user interface design factors that are involved in a multi-device media application could become a valuable future work. On the one hand, these factors will consider the functionality, usability and aesthetic aspects addressed in the literature for single-device interface design. On the other hand, the factors will contemplate the new interface design aspects introduced by the multi-device dimension, not addressed in the literature previously, as one main possible contribution of the future work. Furthermore, another main contribution could be to provide a user interface model for media applications in multiple devices simultaneously, on top of the design factors previously defined. This general model would aim to be as general and flexible as possible, to be able to be parameterised by developers according to the needs of the specific application and over well-known HCI concepts. From these HCI criteria parameterisation, the general model will end on an automatically created set of rules to be applied in a multi-screen system and decide the arbitrary layout templates to be used depending the specific contextual information of the multi-device dimension.

As an example, the application developers or broadcasters will define information in different levels:

- Information about the general layout for different devices, such as which is the most relevant zone of the screen, the relevance of having empty areas in the layout, the impact of having content below the scrollbar, or the importance of having overlaps between elements.
- Information for each one of the elements within the application, defining the relevance of each one of them comparing to the others, as well as the optimal size and aspect ratio, the impact of changing the size or the aspect ratio of the them, or the distance of the element according to the most relevant zone in the screen and its relevance in the application.
- Information about aesthetic parameters, such as the balance, the unity or the sequence.
- Information about multi-screen considerations, such as the affinity of the elements with the end device, particularities of each device, or the number of devices being used at the same time for a user.

Finally, there are other aspects of this research that can be also extended as future work:

- Solutions to provide a more automatic discovery and association of devices on top of technologies such as NFC[1] or Bluetooth[2], and integration of discovering technologies such as DIAL[3], recently incorporated in HbbTV 2.0 version[4].

- Combination of the local network media synchronisation, such as the DVB-CSS[5] included in the recently published HbbTV 2.0 version, and the proposed shared timing solution in order to provide a full hybrid broadcast-Internet synchronisation.

- Extend the use case and scenarios of multi-device media experiences from a home environment focused on the TV as the central device to other scenarios, such as connected cars, or massive events with screens like a music concert.

[1] Near Field Communication: http://nearfieldcommunication.org/

[2] Bluetooth: https://www.bluetooth.com/

[3] DIAL: http://www.dial-multiscreen.org/

[4] HbbTV 2.0 published in February 2015: https://www.hbbtv.org/news-events/hbbtv-2-0-specification-released/

[5] DVB Companion Screen Synchronization: https://www.dvb.org/resources/public/factsheets/dvb-css_factsheet.pdf

# Part IV

# Appendix

APPENDIX

# A

# Other Publications

List of other publications:

1. 
   - **Title:** Reference Model for Hybrid Broadcast Web3D TV
   - **Authors:** Igor García Olaizola, Josu Pérez, Mikel Zorrilla, Angel Martin, Maider Laka
   - **Proceedings:** 3D Web Technology (Web3D)
   - **Pages:** 177-180
   - **Publisher:** ACM
   - **Year:** 2013
   - **DOI:** http://dx.doi.org/10.1145/2466533.2466560
   - **Abstract:** *3DTV can be considered as the biggest technical revolution in TV content creation since the black and white to color transition. However, the big commercial success of current TV market has been produced around the Smart TV concept. Smart TVs connect the TV set to the web and introduce the main home multimedia display in the app world, social networks and content related interactive services. Now, this digital convergence can become the driver for boosting the success of 3DTV industry. In fact, the integration of stereoscopic TV production and Web3D seems to be the next natural step of the hybrid broadband-broadcast services. We propose in this paper a general reference model to allow the convergence of 3DTV and 3D Web by defining a*

*general architecture and some extensions of current Smart TV concepts as well as the related standards.*

---

2. 
- **Title:** HTML5-based System for Interoperable 3D Digital Home Applications
- **Authors:** Mikel Zorrilla, Angel Martin, Jairo R. Sanchez, Iñigo Tamayo, Igor G. Olaizola
- **Proceedings:** Digital Home (ICDH)
- **Pages:** 206-214
- **Publisher:** IEEE
- **Year:** 2012
- **DOI:** http://dx.doi.org/10.1109/ICDH.2012.21
- **Abstract:** *Digital home application market shifts just about every month. This means risk for developers struggling to adapt their applications to several platforms and marketplaces while changing how people experience and use their TVs, smartphones and tablets. New ubiquitous and context-aware experiences through interactive 3D applications on these devices engage users to interact with complex 3D scenes in virtual applications. Interactive 3D applications are boosted by emerging standards such as HTML5 and WebGL removing limitations, and transforming the Web into a horizontal application framework to tackle interoperability over the heterogeneous digital home platforms. Developers can apply their knowledge of web-based solutions to design digital home applications, removing learning curve barriers related to platform-specific APIs. However, constraints to render complex 3D environments are still present especially in home media devices. This paper provides a state-of-the-art survey of current capabilities and limitations of the digital home devices and describes a latency-driven system design based on hybrid remote and local rendering architecture, enhancing the interactive experience of 3D graphics on these thin devices. It supports interactive navigation of sophisticated 3D scenes while provides an interoperable solution that can be deployed over the wide digital home device landscape.*

---

3.
- **Title:** End to end solution for interactive on demand 3d media on home network devices
- **Authors:** Mikel Zorrilla, Angel Martin, Felipe Mogollon, Julen García, Igor G. Olaizola
- **Proceedings:** Broadband Multimedia Systems and Broadcasting (BMSB)
- **Pages:** 1-6
- **Publisher:** IEEE
- **Year:** 2012
- **DOI:** http://dx.doi.org/10.1109/BMSB.2012.6264228
- **Abstract:** *Smart devices have deeply modified the user consumption expectations getting used to rich interactive experiences around new media services. In this emerging landscape, TV rises as the central media device integrating the home network ecosystem. In the race to create more dynamic and customizable content, computer generated 3D graphics get a prominent position combined with video and audio to provide immersive and realistic environments in advanced applications where the user interaction is crucial. However, current home devices lack the required specific hardware to perform it. The proposed 3DMaaS System faces this scenario by performing 3D cloud rendering through streaming sessions with each client device, taking benefit of the Internet connectivity and video streaming management capabilities that most of thin devices have. In order to deal with the wide spectrum of device features, 3DMaaS provides a complete set of streaming formats, including RTSP, HLS and MPEG-DASH, that also fits new trends in media consumption brought by HTML5 and HbbTV. This paper presents latency performance profiling over the different streaming protocols which have a direct influence on the user interaction experience.*

---

4.
- **Title:** HBB4ALL: Deployment of HbbTV services for all
- **Authors:** Pilar Orero, Carlos A. Martín, Mikel Zorrilla
- **Proceedings:** Broadband Multimedia Systems and Broadcasting (BMSB)
- **Pages:** 1-4

- **Publisher:** IEEE
- **Year:** 2015
- **DOI:** http://dx.doi.org/10.1109/BMSB.2015.7177252
- **Abstract:** *Hybrid Broadcast Television for All is a European Commission co-financed project, inside the Competitiveness and Innovation Framework Programme (CIP). The project builds on HbbTV, the European standard for broadcast and broadband multimedia converged services, and looks at how HbbTV technology may be used to enhance access services (such as subtitling, audio description or sign language) on both the production and service sides. HbbTV 1.5 devices are widely available in the market while HbbTV version 2.0 specification has been recently released. TV content can be enhanced by HbbTV applications with additional synchronised services in a personalised manner. For access services this opens an entirely new opportunity for users who may choose an access service delivered via their IP connection which seamlessly integrates with the regular broadcast programme. The presentation will describe the improvements taken on board by HBB4ALL to existing access services and ways of addressing the key technical, organisational and legal obstacles to the sustainable take-up of these services throughout Europe. HBB4ALL focuses on real pilot deployment as a first step to ensure a successful exploitation of these services in a near future. We will offer new insights, from the fields of human machine interaction and social innovation, which arise from the new interactive multimodal and multilanguage services which may be offered. This article will first describe the structure chosen for the project, with four pilots developed in parallel: subtitling, audio description, sign language and user interaction. Then it will describe the methodology and research approaches used for testing the new accessibility services.*

---

5. 
- **Title:** Ontology-based tourism for all recommender and information retrieval system for interactive community displays
- **Authors:** Kevin Alonso, Mikel Zorrilla, Roberto Confalonieri, Javier Vazquez-Salceda, Hasier Iñan, Manel Palau, Javier Calle, Elena Castro

- **Proceedings:** Information Science and Digital Content Technology (ICIDT)
- **Pages:** 650-655
- **Publisher:** IEEE
- **Year:** 2012
- **Abstract:** *This paper presents a multi-stage ontology-based touristic recommender system which offers: personalized suggestions to citizens and tourists, including those with special needs; and information concerning the suggested locations. The system's suggestions are based on user profiles which are continuously updated via feedback obtained from past interactions. Users' preferences are deducted by means of profiles and they are used to create and to send queries to heterogeneous information sources. The results are ranked and presented to the user along with related information.*

---

6.
- **Title:** Next Generation Multimedia on Mobile Devices
- **Authors:** Mikel Zorrilla, M. del Puy Carretero, Alex Ugarte, Felipe Mogollon, David Oyarzun, Igor G. Olaizola
- **Article:** Mobile Technology Consumption: Opportunities and Challenges: Opportunities and Challenges
- **Pages:** 168
- **Publisher:** IGI Global
- **Year:** 2011
- **Abstract:** *The multiplatform consumption of multimedia content has become a crucial factor in the way of watching multimedia. Current technologies such as mobile devices have made people desire access to information from anywhere and at anytime. The sources of the multimedia content are also very important in that consumption. They present the content from many sources distributed on the cloud and mix it with automatically generated virtual reality into any platform. This chapter analyzes the technologies to consume the next generation multimedia and proposes a new architecture to generate and present the content. The goal is to offer it as a service so the users can live the experience in any platform, without requiring any special abilities from*

*the clients. This makes the architecture a very interesting aspect for mobile devices that normally do not have big capabilities of rendering but can benefit of this architecture.*

---

APPENDIX

# B

# Curriculum Vitae

Mikel Zorrilla is with the Department of Digital Media & Broadcasting Technologies, Vicomtech- IK4 since July 2007. He received his Telecommunication Engineering degree in June 2007 from Mondragon Unibertsitatea (Spain) and an advanced degree in Computer Science in June 2012 from UPV/EHU University of Basque Country (Spain). He is the deputy head of the Digital Media & Broadcasting Technologies department and a senior researcher in Interactive Media Technologies. He has been the technical and scientific manager of MediaScape European project [med16] and had participate in several European and local research projects. Currently, he is involved in Hbb4All European project [hbb16] where accessibility for media services from Connected TVs is addressed.

In 2014 he has been an associate professor in the double degree MBA (Master in Business Administration) & Computer Engineering in Deusto Business School in the field of Media Technologies.

Before joining Vicomtech-IK4, he has held positions at Ikerlan S. Coop. as an assistant researcher (2002-2007) in the field of Transport of Multimedia Traffic.

# Part V

# Bibliography

# Bibliography


[ABN13] Ingar M Arntzen, Njål T Borch, and Christopher P Needham. The media state vector: A unifying concept for multi-device media navigation. In *Proceedings of the 5th Workshop on Mobile Video*, pages 61–66. ACM, 2013. 26, 33, 50

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010. 147

[AKA+12] Eero Aho, Kimmo Kuusilinna, Tomi Aarnio, Janne Pietiainen, and Jari Nikara. Towards real-time applications in mobile web browsers. In *Embedded Systems for Real-time Multimedia (ESTIMedia), 2012 IEEE 10th Symposium on*, pages 57–66. IEEE, 2012. 152

[AL11] Ahamed Altaboli and Yingzi Lin. Investigating effects of screen layout elements on interface and screen design aesthetics. *Advances in Human-Computer Interaction*, 2011:5, 2011. 181

[AM06] Sanjay P Ahuja and Jack R Myers. A survey on wireless grid computing. *The Journal of Supercomputing*, 37(1):3–21, 2006. 143, 147

[ama] Amazon Web Services (AWS). http://aws.amazon.com. [Online; accessed 26-January-2016]. 149

[Amd67] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM. 163

[An 13] An Ericsson Consumer Insight Summary Report. TV and Media. Identifying the needs of tomorrow's video consumer. Ericsson Consum-erLab. http://www.ericsson.com/res/docs/2013/consumerlab/tv-and-media-consumerlab2013.pdf, 2013. 7, 27

[Ant12] Gary Anthes. Html5 leads a web revolution. *Communications of the ACM*, 55(7):16–17, 2012. 3, 22, 55

[ASMT11] Matti Anttonen, Arto Salminen, Tommi Mikkonen, and Antero Taivalsaari. Transforming the web into a real application platform: new technologies, emerging trends and missing pieces. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 800–807. ACM, 2011. 107, 125, 144, 149

[Azu97] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385, 1997. 110

[BJM08] Adam Barth, Collin Jackson, and John C Mitchell. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 75–88. ACM, 2008. 161

[BL08] Michael Bauerly and Yili Liu. Effects of symmetry and number of compositional elements on interface and design aesthetics. *Intl. Journal of Human–Computer Interaction*, 24(3):275–287, 2008. 181

[But11] Margaret Butler. Android: Changing the mobile landscape. *IEEE Pervasive Computing*, 10(1):4–7, 2011. 105

[BV13] R van Brandenburg and AT Veenhuizen. Immersive second-screen experiences using hybrid media synchronization. 2013. 26, 52

[can] CanIUse.com Support of WebGL in different Web Browsers. http://caniuse.com/. [Online; accessed 26-January-2016]. 162

[CB13] F.Ozgur Catak and M.Erdal Balaban. Cloudsvm: Training an svm classifier in cloud computing systems. 7719:57–68, 2013. 127, 150

[CBJ08] Pablo Cesar, Dick CA Bulterman, and AJ Jansen. Usages of the secondary screen in an interactive television environment: Control, enrich, share, and transfer television content. In *Changing television environments*, pages 168–177. Springer, 2008. 3, 23, 180

[CD12] Cédric Courtois and Evelien D'heer. Second screen applications and tablet users: constellation, awareness, experience, and interest. In *Proceedings of the 10th European conference on Interactive tv and video*, pages 153–156. ACM, 2012. 3, 23, 55

[CDM] Cloud Data Management Interface (CDMI). http://www.snia.org/cdmi. [Online; accessed 26-January-2016]. 149

[CIM] Cloud Infrastructure Management Interface (CIMI). http://dmtf.org/standards/cloud. [Online; accessed 26-January-2016]. 149

[CK10] Marco Conti and Mohan Kumar. Opportunities in opportunistic computing. *Computer*, 43(1):42–50, 2010. 146

[CKP+93] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. Logp: towards a realistic model of parallel computation. *SIGPLAN Not.*, 28(7):1–12, July 1993. 135

[Cla12] Lynn Claudy. The broadcast empire strikes back. *Spectrum, IEEE*, 49(12):52–58, 2012. 3, 6, 23, 26

[CPK+13] Reginald Cushing, Ganeshwara Herawan Hananda Putra, Spiros Koulouzis, Adam Belloum, Marian Bubak, and Cees De Laat. Distributed computing on an ensemble of browsers. *Internet Computing, IEEE*, 17(5):54–61, 2013. 150

[cud07] CUDA computing platform and programming model. http://www.nvidia.com/object/cuda_home_new.html, Feb 2007. [Online; accessed 26-January-2016]. 151

[CWH13] Aniruddha Chandra, Jon Weissman, and Benjamin Heintz. Decentralized edge clouds. *Internet Computing, IEEE*, 17(5):70–73, 2013. 149, 150

[DA09] Petros Daras and Federico Alvarez. A future perspective on the 3d media internet. In *Future Internet assembly*, pages 303–312, 2009. 4, 110

[DFMGS11] Gianmarco De Francisci Morales, Aristides Gionis, and Mauro Sozio. Social content matching in mapreduce. *Proc. VLDB Endow.*, 4(7):460–469, April 2011. 127, 150

[DIB+16] Heiko Desruelle, Simon Isenberg, Andreas Botsikas, Paolo Vergori, and Frank Gielen. Accessible user interface support for multi-device ubiquitous applications: architectural modifiability considerations. *Universal Access in the Information Society*, 15(1):5–19, 2016. 26

[DLIG12] Heiko Desruelle, John Lyle, Simon Isenberg, and Frank Gielen. On the challenges of building a web-based ubiquitous application platform. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 733–736. ACM, 2012. 77

[DSML+11] Robbie De Sutter, Mike Matton, Niels Laukens, Dieter Van Rijsselbergen, and Rik Van de Walle. Mediacrm: enabling customer relationship management in the broadcast. In *International Conference on Web Information Systems and Mining*, pages 19–26. Springer, 2011. 73

[DSN] Robbie De Sutter and Lode Nachtergaele. Crm. 73

[EBU14] EBU Recommendation: EBU Tech 3366. TCP-CPA: Cross-Platform Authentication. https://tech.ebu.ch/docs/tech/tech3366.pdf, 2014. 26

[Eye12] EyeALike similarity across large datasets. http://www.eyealike.com/, 2012. [Online; accessed 26-January-2016]. 150

[Far11] Stephen Farrell. Leaky or guessable session identifiers. *IEEE Internet Computing*, 15(1):88–91, 2011. 77

[FE10] Philipp Fechteler and Peter Eisert. Accelerated video encoding using render context information. In *2010 IEEE International Conference on Image Processing*, pages 2033–2036. IEEE, 2010. 115

[Fet11] Ian Fette. The websocket protocol. 2011. 130, 144, 158

[FKT01] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance computing applications*, 15(3):200–222, 2001. 146

[FW78] Steven Fortune and James Wyllie. Parallelism in random access machines. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 114–118. ACM, 1978. 135, 165

[Gar05] J. J. Garrett. Ajax: A new approach to web applications. 2005. 131, 144, 158

[Gar14] Gartner report. Forecast: PCs, Ultramobiles, and Mobile Phones, Worldwide, 2010-2017. http://www.gartner.com/newsroom/id/2645115, 2014. 7

[Gar17] Gartner report. Forecast: PCs, Ultramobiles, and Mobile Phones, Worldwide. http://www.gartner.com/newsroom/id/2645115, 2010-2017. 27

[GLG11] Carl A Gutwin, Michael Lippold, and TC Graham. Real-time groupware in the browser: testing the performance of web-based networking. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 167–176. ACM, 2011. 78

[gum12] Gum Gum in-image advertising platform. http://gumgum.com/, 2012. [Online; accessed 26-January-2016]. 150

[had05] Apache Hadoop framework. http://hadoop.apache.org/, 2005. [Online; accessed 26-January-2016]. 150

[hbb16] Hbb4All EU project. http://hbb4all.eu/, 2013-2016. 193

[HEB+01] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordan Stoll, Matthew Everett, and Pat Hanrahan. Wiregl: a scalable graphics system for clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 129–140. ACM, 2001. 116

[HFF11] Ingo Hofmann, Nikolaus Färber, and Harald Fuchs. A study of network performance with application to adaptive http streaming. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium on*, pages 1–6. IEEE, 2011. 118

[HHLD11] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011. 152

[HHN+02] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D Kirchner, and James T Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM transactions on graphics (TOG)*, 21(3):693–702, 2002. 116

[hip12] Hadoop Image Processing Interface. http://hipi.cs.virginia.edu/, 2012. [Online; accessed 26-January-2016]. 150

[HXW13] Dijiang Huang, Tianyi Xing, and Huijun Wu. Mobile cloud computing service models: a user-centric approach. *Network, IEEE*, 27(5):6–11, 2013. 143, 147

[JBG12] Won Jeon, Tasneem Brutch, and Simon Gibbs. Webcl for hardware-accelerated web applications. In *TIZEN Developer Conference May*, pages 7–9, 2012. 144, 152

[JLN12] Sverre Jarp, Alfio Lazzaro, and Andrzej Nowak. The future of commodity computing and many-core versus the interests of hep software. In *Journal of Physics: Conference Series*, volume 396, page 052058. IOP Publishing, 2012. 151

[JMS05] François Jammes, Antoine Mensch, and Harm Smit. Service-oriented device communications using the devices profile for web services. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8. ACM, 2005. 86

[jsm13] JSMapReduce JS library. http://jsmapreduce.com/, 2013. [Online; accessed 26-January-2016]. 150

[Kal12] Kalooga discovery service for image galleries. http://www.kalooga.com/, 2012. [Online; accessed 26-January-2016]. 150

[LC11] Mark Lochrie and Paul Coulton. Mobile phones as second screen for tv, enabling inter-audience interaction. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, page 73. ACM, 2011. 72

[Leg12] LegitReviews. Google Nexus 7 Tablet Review. http://www.legitreviews.com/article/1988/2/, Jul 2012. [Online; accessed 26-January-2016]. 165

[LG10] Peter Lubbers and Frank Greco. Html5 web sockets: A quantum leap in scalability for the web. *SOA World Magazine*, (1), 2010. 78

[Liu13] Huan Liu. Big data drives cloud adoption in enterprise. *IEEE internet computing*, (4):68–71, 2013. 147

[LLJ+09] Arto Laikari, Jukka-Pekka Laulajainen, Audrius Jurgelionis, Philipp Fechteler, and Francesco Bellotti. Gaming platform for running games on low-end devices. In *International Conference on User Centric Media*, pages 259–262. Springer, 2009. 115

[LMA+10] Heshan Lin, Xiaosong Ma, Jeremy Archuleta, Wu-chun Feng, Mark Gardner, and Zhe Zhang. Moon: Mapreduce on opportunistic environments. pages 95–106, 2010. 127, 150

[LNIF13] John Lyle, Claes Nilsson, Anders Isberg, and Shamal Faily. Extending the web to support personal network services. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 711–716. ACM, 2013. 86

[LS07] Fabrizio Lamberti and Andrea Sanna. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE transactions on visualization and computer graphics*, 13(2):247–260, 2007. 116

[LWB13] Philipp Langhans, Christoph Wieser, and François Bry. Crowdsourcing mapreduce: Jsmapreduce. pages 253–256, 2013. 127, 150

[MC13] Tommy MacWilliam and Cris Cecka. Crowdcl: Web-based volunteer computing with webcl. In *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–6. IEEE, 2013. 152

[Med14] MediaScape EU project report. Usage Scenarios and Requirements. http://mediascapeproject.eu/files/D2.1.pdf, 2014. 6, 26, 34, 35

[med16] MediaScape EU project. http://mediascapeproject.eu, 2013-2016. 193

[Mer10] Klaus Merkel. Hbbtv—a hybrid broadcast-broadband system for the living room. *EBU technical review-2010 Q*, 1, 2010. 72

[Mer11] Klaus Merkel. Hybrid broadcast broadband tv, the new way to a comprehensive tv experience. In *Electronic Media Technology (CEMT), 2011 14th ITG Conference on*, pages 1–4. IEEE, 2011. 4, 23, 29, 55, 72, 85

[MGV+10] Giuseppe Marino, Paolo Simone Gasparello, Davide Vercelli, Franco Tecchia, and Massimo Bergamasco. Network streaming of dynamic 3d content with on-line compression of frame data. In *2010 IEEE Virtual Reality Conference (VR)*, pages 285–286. IEEE, 2010. 116

[Mil14] Millward Brown. Marketing in a Multi-screen World. AdReaction, 2014. 3, 6, 7, 27

[MK06] Niamh McNamara and Jurek Kirakowski. Functionality, usability, and user experience: three areas of concern. *interactions*, 13(6):26–28, 2006. 181

[Mob13] Per Moberg. Event-driven interactivity in application-based tv-programs. 2013. 78

[mpi96] MPI: Message Passing Interface. http://www.mcs.anl.gov/research/projects/mpi/, 1996. [Online; accessed 26-January-2016]. 151

[MSS02] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: multimedia content description interface*, volume 1. John Wiley & Sons, 2002. 30

[MTCK11] Abedelaziz Mohaisen, Huy Tran, Abhishek Chandra, and Yongdae Kim. Socialcloud: Using social networks for building distributed computing services. *CoRR*, abs/1112.2254, 2011. 127

[MTP12] Christina Moreno, Nicolas Tizon, and Marius Preda. Mobile cloud convergence in gaas: a business model proposition. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1344–1352. IEEE, 2012. 115

[MVML13] Rafael Moreno-Vozmediano, Rubén S Montero, and Ignacio M Llorente. Key challenges in cloud computing: Enabling the future internet of services. *Internet Computing, IEEE*, 17(4):18–25, 2013. 149

[NAP14a] NAPTE. New NAPTE and CEA research finds show producers and creators see second screen becoming permanent part of viewing experience. https://www.natpe.com/press/release/130, Q1 2014. [Online; accessed 26-January-2016]. 153

[NAP14b] NAPTE survey. Study on Second Screen. https://www.natpe.com/press/release/130, 2014. 7, 28

[NBRK11] Dirk Neumann, Christian Bodenstein, Omer F Rana, and Ruby Krishnaswamy. Stacee: enhancing storage clouds using edge devices. In *Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics*, pages 19–26. ACM, 2011. 143, 147

[NCB06] Daniele Nadalutti, Luca Chittaro, and Fabio Buttussi. Rendering of x3d content on mobile devices with opengl es. In *Proceedings of the eleventh international conference on 3D web technology*, pages 19–26. ACM, 2006. 116

[Nie14a] Nielsen report. Shifts in viewing: The cross platform report. http://www.nielsen.com/us/en/insights/reports/2014/shifts-in-viewing-the-cross-platform-report-q2-2014.html, 2014. 6, 27

[Nie14b] Nielsen report. State of the Media: Audio today. http://www.nielsen.com/us/en/insights/reports/2014/state-of-the-media-audio-today-q3-2014.html, 2014. 6, 27

[Nie14c] Nielsen survey. Sports Fans Amplify The Action Across Screens. http://www.nielsensocial.com/sports-fans-amplify-the-action-across-screens/, 2014. 7, 28, 153

[Nie14d] Nielsen survey. Who's tweeting about tv? http://www.nielsensocial.com/whos-tweeting-about-tv/, 2014. 7, 28, 153

[Nok] NokiaResearch. Nokia WebCL Extension for Firefox. https://github.com/toaarnio/webcl-firefox. [Online; accessed 26-January-2016]. 162

[NWG+09] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009. 147

[OCC] Open Cloud Computing Interface (OCCI). http://occi-wg.org. [Online; accessed 26-January-2016]. 149

[OJ10] Sixto Ortiz Jr. Is 3 d finally ready for the web? *Computer*, 43(1):14–16, 2010. 109

[ope] OpenNebula Project. http://opennebula.org/. [Online; accessed 26-January-2016]. 149

[ope12] OpenNM Specification. http://www.khronos.org/registry/cl/, Nov 2012. [Online; accessed 26-January-2016]. 148

[ope13] OpenNM Specification. http://openmp.org/wp/openmp-specifications/, Mar 2013. [Online; accessed 26-January-2016]. 151

[OQFS14] Igor G Olaizola, Marco Quartulli, Johana Florez, and Basilio Sierra. Trace transform based method for color image domain identification. *Multimedia, IEEE Transactions on*, 16(3):679–685, 2014. 162

[Ort11] Sixto Ortiz. Bringing 3d to the small screen. *Computer*, 44(10):11–13, 2011. 110

[PE02] Fernando CN Pereira and Touradj Ebrahimi. *The MPEG-4 book*. Prentice Hall Professional, 2002. 30

[Pes] Kamila Peszko. Multiscreening as a way of reaching a consumer–idea and possibilities. 23

[RLMT12] Benjamin Rainer, Stefan Lederer, Christopher Müller, and Christian Timmerer. A seamless web integration of adaptive http streaming. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1519–1523. IEEE, 2012. 118

[SCMM09] Luiz Fernando Gomes Soares, Romualdo MR Costa, Marcio Ferreira Moreno, and Marcelo F Moreno. Multiple exhibition devices in dtv systems. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 281–290. ACM, 2009. 29

[SDTDD09] Pieter Simoens, Filip De Turck, Bart Dhoedt, and Piet Demeester. Remote display solutions for mobile cloud computing. *IEEE Internet Computing*, 13(5):46–53, 2009. 116

[set99] SETI@home Project. `http://setiathome.berkeley.edu/`, May 1999. [Online; accessed 26-January-2016]. 146

[SGS10] John E Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(1-3):66–73, 2010. 148

[SLLE10] Weidong Shi, Yang Lu, Zhu Li, and Jonathan Engelsma. Scalable support for 3d graphics applications in cloud. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 346–353. IEEE, 2010. 116

[SMNM10] Luiz Fernando Gomes Soares, Marcio Ferreira Moreno, Carlos De Salles Soares Neto, and Marcelo Ferreira Moreno. Ginga-ncl: declarative middleware for multimedia iptv services. *IEEE Communications Magazine*, 48(6):74–81, 2010. 29

[str] Strategy Analytics. https://www.strategyanalytics.com/. [Online; accessed 26-January-2016]. 165

[Str14] Strategy Analytics. 2013 Smart TV Shipments Grew 55 Percent. http://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5472, 2014. 6, 27

[Swe11] Chris Sweeney. Hipi: A hadoop image processing interface for image-based mapreduce tasks. *B.S. Thesis. University of Virginia, Department of Computer Science 2011.*, 2011. 127, 150

[TBSD13] Wei Tan, M Brian Blake, Iman Saleh, and Schahram Dustdar. Social-network-sourced big data analytics. *IEEE Internet Computing*, (5):62–69, 2013. 152

[Tec13] Techrunch. Google Believes Web Components Are The Future Of Web Development. http://tcrn.ch/2a1ZZpN, 2013. 33, 58

[TFPK13] Andrew Turner, Andrew Fox, John Payne, and Hyong S Kim. C-mart: Benchmarking the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1256–1266, 2013. 77

[TKBS14] IVishal M Toshniwal, P Kawale, L Bhanage, and S Sonawane. Virtualized screen: A third element for cloud-mobile convergence. 2014. 116

[TL10] Sasu Tarkoma and Eemil Lagerspetz. Arching over the mobile chasm: Platforms and runtimes. *Computer*, 2010. 102, 105

[TM11] Antero Taivalsaari and Tommi Mikkonen. The web as an application platform: The saga continues. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 170–174. IEEE, 2011. 107, 125

[TMAS11] Antero Taivalsaari, Tommi Mikkonen, Matti Anttonen, and Arto Salminen. The death of binary software: End user software moves to the web. In *2011 Ninth international conference on creating, connecting and collaborating through computing*, pages 17–23. IEEE, 2011. 107, 125

[Tom11] TomsHardware. AMD Bulldozer Review: FX-8150 Gets Tested. http://www.tomshardware.com/reviews/fx-8150-zambezi-bulldozer-990fx,3043-14.html, Oct 2011. [Online; accessed 26-January-2016]. 165

[TV10] Stefan Tilkov and Steve Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80, 2010. 74, 150

[Val90] Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990. 136, 165

[VM15] Radu-Daniel Vatavu and Matei Mancas. Evaluating visual attention for multi-screen television: measures, toolkit, and experimental findings. *Personal and Ubiquitous Computing*, 19(5-6):781–801, 2015. 180

[W3C08] W3C World-Wide Web Consortium. Synchronized Multimedia Integration Language – SMIL 3.0 Specification, W3C Recommendation. http://www.w3.org/TR/2008/REC-SMIL3-20081201/, 2008. 30

[W3C10] W3C Recommendation. Mobile Web Application Best Practices. http://www.w3.org/TR/mwabp/, 2010. 31, 58

[W3C12] W3C Recommendation. Media Queries. http://www.w3.org/TR/css3-mediaqueries/, 2012. 31, 58

[W3C14a] W3C Editor's Draft. HTML 5.1 Nightly. A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/html/wg/drafts/html/master/embedded-content.html#the-source-element-when-used-with-the-picture-element, 2014. 31, 58

[W3C14b] W3C Editor's Draft. Media Queries Level 4. http://dev.w3.org/csswg/mediaqueries4/#script-custom-mq, 2014. 46

[W3C14c] W3C Working Draf . CSS Flexible Box Layout Module Level 1. http://www.w3.org/TR/css3-flexbox/, 2014. 32, 58

[W3C14d] W3C Working Draft. CSS Grid Layout Module Level 1. http://www.w3.org/TR/css3-grid-layout/, 2014. 32, 58

[W3C14e] W3C Working Draft. CSS Regions Module Level 1. http://www.w3.org/TR/css3-flexbox/, 2014. 32, 58

[W3C14f] W3C Working Group Note. Introduction to Web Components. http://w3c.github.io/webcomponents/explainer/, 2014. 32, 57

[W3C15a] W3C Community Group. Multi-Device Timing Community Group. https://www.w3.org/community/webtiming/, 2015. 26, 33, 50

[W3C15b] W3C Editor's Draft. Timing Object. http://webtiming.github.io/timingobject/#broadcasting---time-consistent-live-web-productions, 2015. 52

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991. 6, 26

[WJW13] Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. Plug into the supercloud. *Internet Computing, IEEE*, 17(2):28–34, 2013. 147

[YHL11] Chao-Tung Yang, Chih-Lin Huang, and Cheng-Fang Lin. Hybrid cuda, openmp, and mpi parallel programming on multicore gpu clusters. *Computer Physics Communications*, 182(1):266–269, 2011. 151

[ZBD+15] Mikel Zorrilla, Njål Borch, François Daoust, Alexander Erk, Julián Flórez, and Alberto Lafuente. A web-based distributed architecture for multi-device adaptation in media applications. *Personal and Ubiquitous Computing*, 19(5-6):803–820, 2015. 154

[ZDLB08] Theodore Zahariadis, Petros Daras, and Isidro Laso-Ballesteros. Towards future 3d media internet. *NEM Summit*, pages 13–15, 2008. 4, 110

[ZdPCU+11] Mikel Zorrilla, María del Puy Carretero, Alejandro Ugarte, Juan Felipe Mogollón, David Oyarzun, and Igor García Olaizola. Next generation multimedia on mobile devices. *Mobile Technology Consumption: Opportunities and Challenges: Opportunities and Challenges*, page 168, 2011. 116

[Zie13] Christoph Ziegler. Second screen for hbbtv—automatic application launch and app-to-app communication enabling novel tv programme related second-screen scenarios. In *Consumer Electronics?? Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on*, pages 1–5. IEEE, 2013. 4, 23, 26, 51, 56, 89, 90

[ZMM+12] Mikel Zorrilla, Ángel Martín, Felipe Mogollón, Julen García, and Igor G Olaizola. End to end solution for interactive on demand 3d media on home network devices. In *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–6. IEEE, 2012. 118, 119

[ZMN05] Fen Zhu, Matt W Mutka, and Lionel M Ni. Service discovery in pervasive computing environments. *IEEE Pervasive computing*, 4(4):81–90, 2005. 87

[ZMS+13] Mikel Zorrilla, Angel Martin, JairoR. Sanchez, Iñigo Tamayo, and IgorG. Olaizola. Html5-based system for interoperable 3d digital home applications. *Multimedia Tools and Applications*, pages 1–21, 2013. 125

[ZMT+13] Mikel Zorrilla, Andrew Martin, Inigo Tamayo, Naiara Aginako, and Igor G Olaizola. Web browser-based social distributed computing platform applied to image analysis. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 389–396. IEEE, 2013. 146, 155, 165, 166, 167

[ZMT+14] Mikel Zorrilla, Angel Martin, Iñigo Tamayo, Sean O'Halpin, and Dominique Hazael-Massieux. Reaching devices around an hbbtv television. In *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–7. IEEE, 2014. 26

[ZTMD15] Mikel Zorrilla, Iñigo Tamayo, Angel Martin, and Ana Dominguez. User interface adaptation for multi-device web-based media applications. In *2015 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–7. IEEE, 2015. 45

[ZTMO13] Mikel Zorrilla, Iñigo Tamayo, Angel Martin, and Igor G Olaizola. Cloud session maintenance to synchronise hbbtv applications and home network devices. In *2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2013. 23, 26, 27, 51, 56, 89, 90

[ZWZ+13] Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R Lyu, and Jianmin Wang. Qos ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1213–1222, 2013. 77