End to end solution for interactive on demand 3D Media on home network devices

Mikel Zorrilla, Ángel Martín, Felipe Mogollón, Julen García and Igor G. Olaizola

***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.**

***Index Terms*—Media streaming, multimedia broadcasting, remote rendering**

I. Introduction

IN terms of multimedia content the user consumption trends are constantly changing fuelled by new technologies. Television is still the main device for watching media content at home. Nevertheless in the same way that mobile phones have gone from thin to smart, 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". It allows accessing to customized on demand content and interactive services fostering active attitude of the users. On the other hand, users are rapidly getting used to features coming from Smart TVs, smartphones and tablets transforming the viewing experience into a more interactive one. New media platforms are being increasingly oriented to access device-independent content as a service to deal with the home network device landscape.

Not only the usage of multimedia consumption has changed, multimedia content itself has also suffered a great evolution. The embracement of new technologies such as computer generated 3D, allows creating attractive experiences by combining rich media. This is known as 3D Media [1], [2] and it is composed of different audio and video sources, images and computer generated 3D objects.

Interaction rises as fundamental to handle 3D Media content. Real-time computer generated 3D technologies provide the user a complete immersion and personalization in holistic environments. In this context, the interaction latency of the system is a crucial factor to guarantee the Quality of Experience (QoE) of the user.

All these features need advanced graphic capabilities and the current home network devices do not have such a powerful specific hardware to render the complex 3D Media content that users are demanding. In contrast, these devices are very suitable for this kind of applications that deal with great detail and realistic immersive 3D scenes combined with movies and soundtracks. That's why a solution to overcome this hurdle is needed.

3DMaaS System (3D Media as a Service System) is an end to end solution able to extend the multimedia consumption capabilities of the home network devices. *3DMaaS* enables the consumption of 3D Media content through any device using client-server architecture for remote rendering. The system pushes the required processing to the cloud and it offers different output streaming protocols with the cloud-rendered content to adapt it to the capabilities of the client-device and to offer the best low-latency option in each case.

Both the proprietary application frameworks and the standards are ready for video stream reception in applications

Manuscript received April 27, 2012. This work was supported in part by the Spanish Ministry for Industry, Tourism and Commerce, and the European FEDER, funding the project where these results have been developed.

M. Zorrilla is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net, Donostia-San Sebastián, 20009 Spain (phone: +34-943-309-230; fax: +34-943-309-393; e-mail: mzorrilla@vicomtech.org).

A. Martin is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net, Donostia-San Sebastián, 20009 Spain (e-mail: amartin@vicomtech.org).

F. Mogollon is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net, Donostia-San Sebastián, 20009 Spain (e-mail: fmogollon@vicomtech.org).

J. García is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net, Donostia-San Sebastián, 20009 Spain (e-mail: jgarcia@vicomtech.org).

I. Olaizola is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net, Donostia-San Sebastián, 20009 Spain (e-mail: jgarcia@vicomtech.org).

through home network devices. On the one hand, HTML5[1] provides an interoperable platform [3], [4] to develop applications for smartphones, tablets and PCs. RTSP or Dynamic Adaptive Streaming over HTTP can be supported depending on the browser implementation. Browsers such as Google Chrome, Opera, Safari or Firefox bet on Adaptive HTTP Streaming including it on their development roadmaps. On the other hand, HbbTV 1.5 specification[2] supports MPEG-DASH for applications through TVs and set-top boxes.

These technologies provide adaptation mechanisms to accommodate the dynamic behaviour of the content to the decisions of the user and fit the media delivery protocols for the context. This way the user can consume on demand seamless device-adapted 3D Media content on home network devices.

Section 2 presents a state of the art of the different video streaming protocols. In Section 3 there is an overall overview of *3DMaaS System* description and its benefits regarding the state of the art. Section 4 shows the validation experiments done with *3DMaaS System* to assess the variation of the latency and jitter parameters through the different output streaming protocols. Results and conclusions are on Section 5.

II. State of the Art

This section focuses on the state of the art of the different media streaming approaches. When it comes to real-time media delivery, the emphasis lays on low latency and jitter, and efficient transmission. Although current media streaming protocols differ in implementation details, they can be classified into two main categories: push-based and pull-based protocols [5], [6].

Push-based streaming protocols establish a connection procedure at the communication beginning and the server streams the media content to the client until it stops the session. Real-time Streaming Protocol (RTSP), specified in RFC 2326, is the most deployed push-based streaming protocol. It is performed on top of Real-time Transport Protocol (RTP), specified in RFC 3550. RTP runs over UDP where the streaming parameters, such as bitrate, are managed by the application. RTP is oriented to a best-effort media transmission for applications that require low-latency.

On the contrary, in pull-based streaming protocols the streaming client requests media content to the server in an active way. HTTP is the most common protocol for pull-based media delivery. HTTP represents a widely supported protocol, and provides authentication and authorization infrastructure, a better NAT/Firewall transversal capability and exploits existing HTTP infrastructure (cheaper CDNs, etc.).

Nowadays, progressive download is the most extended pull-based media streaming protocol. In this approach, the media client sends an HTTP request to the server, and it starts pulling the content on a best-effort way. The client waits for a minimum required buffered level that depends on its implementation, and starts playing it while the progressive download continues in the background. However, progressive download does not offer the flexibility and rich features of streaming and it has three main disadvantages: (1) wasteful of bandwidth if the user stops watching the content, since data have been transferred, (2) scalability, since there is no bit-rate adaptation and, (3) no support for live or real-time media [7].

Dynamic Adaptive Streaming over HTTP [8] is an hybrid of progressive download and traditional streaming, and it is being positioned as the on-demand/live/real-time streaming approach for home network media representation standards such as HTML5 and HbbTV. It is a pull-based protocol where the client sends HTTP requests to download specific segments of the content from an HTTP server and the content is played while the next segments are being buffered. The duration of each segment is short, enabling the client to download only the necessary content and simulating a full streaming protocol. It provides a better adaptation to dynamic conditions and device capabilities. Therefore each segment can be fitted for dynamic conditions through Internet and/or home network, display resolution, CPU and memory resources, etc. Adaptive streaming enables a better Quality of Experience (QoE) comparing to progressive download with faster start-up and seeking, quicker buffer fills, etc. It also benefits from the entire HTTP protocol infrastructure comparing with push-based streaming protocols such us RTSP, but it has a dramatic impact on the latency, which can be critical for real-time interactive applications.

Adaptive media streaming protocols have a manifest with a list of accessible segments and their timing, corresponding to different resolutions and quality levels, so the client can switch between different bitrates according to changeable context. The client player requests the most suitable segment and the server updates dynamically the manifest on a live streaming environment.

There are different proprietary implementations of adaptive streaming protocols such as Microsoft Smooth Streaming, Apple HTTP Live Streaming (HLS) or Adobe HTTP Dynamic Streaming. In November 2011 MPEG-DASH had been accepted by ISO as an International Standard for Dynamic Adaptive Streaming over HTTP with the purpose to converge all the proprietary approaches into the standard.

Microsoft Smooth Streaming[3] is based on Protected Interoperable File Format (PIFF), an extension of the MPEG4 Part 12 (MP4) specification. All the same bitrate segments are packed in a MP4 file and segments are usually two seconds long. The client has to download a client-side manifest to know the available segments, codecs, video resolutions, bitrates, etc. The client uses that information to make HTTP requests to the server asking for specific segments according to the context. The server checks in a server-side manifest in which MP4 is the segment, and adds the stream related metadata to the media content.

[1] HTML5 standard specification (May2011) http://www.w3.org/TR/html5/
[2] HbbTV 1.5 specification (April 2012) http://www.hbbtv.org
[3] http://www.iis.net/download/SmoothStreaming

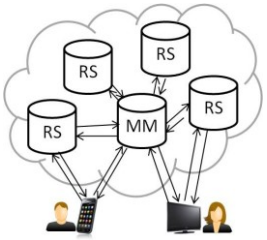

Fig. 1. General diagram of the *3DMaaS System*

Apple HTTP Live Streaming[4] is based on ISO/IEC 13818-1 MPEG2 Transport Stream file format for the media data storage. Each segment is stored as a different transport stream file unlike Smooth Stream. There is also a client-side manifest based on an extension of the MP3 playlist file standard. The client adds different segments creating a playlist dynamically. The manifest can be updated for live and real-time streaming applications.

Adobe HTTP Live Streaming[5] uses fragmented MP4, as Smooth Streaming does. It enables on-demand and live streaming and supports HTTP and Real Time Messaging Protocol (RTMP). The media file specification F4V is based on MPEG-4 Part 12, and the client-side manifest is specified as a F4M (Flash Media Manifest).

MPEG-DASH (ISO/IEC 23009-1) is the first International Standard for dynamic adaptive media streaming over HTTP, based on the previous work of the 3GPP group [9]. It defines the MPD (Media Presentation Description) manifest and delivery formats using ISO BMFF and MPEG2-TS. Depending on the implementation of the client it can flexibly decide when and how download each media segment to build the adapted media content. It supports live [10], on-demand and time-shifted content delivery, advertisement insertion, DRM [11], etc. The MPD file is an XML-based schema which contains the redundant information of the content, such as the content role, codec, DRM, language, resolution, etc. It also has access and timing information, an HTTP URL of each media segment and the earliest next update of the MPD on the server for live streaming.

Next section presents *3DMaaS System* which provides different Adaptive HTTP Streaming protocols that benefit from the already existing HTTP infrastructure making it accessible for most of the home network devices. However, the system provides also an RTSP stream output which enables a low-latency approach for the devices that support it.

III. 3DMAAS SYSTEM DESCRIPTION

3DMaaS System is composed by three main actors (Fig. 1) [12]: MaaS Manager (MM), Rendering Server (RS) and end devices. The role of the MM is to manage the service requests. The system will provide a video stream containing 3D composition to end devices. MM makes a preliminary negotiation with the end-client to evaluate context and its needs.

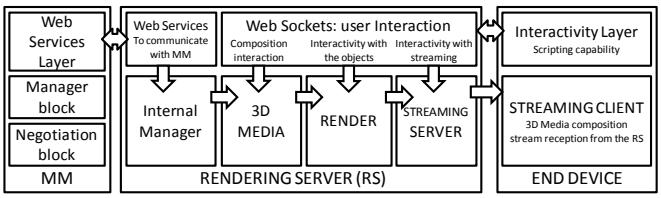

Fig. 2. The block diagram of MM and RS and their communications with the end device

MM establishes communication with available RSs depending on the previous negotiation, and evaluates which one has enough free rendering resources to respond the received request. The entire negotiation context is transferred from the MM to the RS and from that moment on the communication is established between the end device and the RS directly where RS provides a real-time continuous video stream to the end device.

The composition can be produced by the combination of audio, video, images, and 3D objects. The system enables the user to interact with each element of the scene and modify it in real time sending action events through web sockets captured by the RS.

RS renders the customized 3D Media content for each client and it also codifies the content in real-time for the delivery of a video stream to the end device. The user is not only able to interact with the 3D Media content itself but depending on its context (device, bandwidth, etc.) can also adjust the streaming protocol and the coding parameters of the stream. This way, *3DMaaS* provides an adaptable content delivery depending on the features of the device using different streaming protocols such as RTSP, HLS, MPEG-DASH or Icecast[6]. Therefore the features that *3DMaaS System* requires for the 3D Media reception are really affordable for any kind of Connected TV, set-top box, smartphone or tablet.

A block diagram of *3DMaaS System* is shown in Fig. 2 and all the modules are more deeply explained below.

A. Maas MANAGER (MM)

This module holds the system whole management. MM makes the first evaluation and negotiation with the end device and establishes the communication between the RS and the end-client for the stream delivery. It has 3 different blocks:

Negotiation block: It analyzes the 3D Media content description and connection context provided by the end-client. According to this information it decides the streaming protocol, codec parameters and 3D composition.

Manager block: During the negotiation, the computational load required at the server side is also estimated. Depending on this, MM decides what RS may offer the service. MM reports the entire context and the decisions taken to the chosen RS. A direct communication between the end-device and the RS is bridged regarding this information, and the 3D Media stream starts.

[4] https://developer.apple.com/resources/http-streaming/
[5] http://www.adobe.com/products/hds-dynamic-streaming.html
[6] http://www.icecast.org/

Web Services layer: It is aimed for communication with the final client to the negotiation of the initial request, and to transfer the RS the connection context already established with the end device.

B. RENDERING SERVER (RS)

This module creates the 3D Media composition, encoding it in real-time and delivering it as a video stream to the end device. It supports the interaction of the final client both with the content and the parameters of the stream that adapt the needs of the context through web sockets. Each RS is able to manage multiple specific clients depending on the computational capability of the RS.

3DMaaS offers a system that manages a resource pool to achieve scalability for a target QoS. Each RS is unaware of the rest of them and focuses on handling the requests made by MM, who is responsible of continuously monitoring the load state of each RS and performing load balancing strategies.

Fig. 2 shows the different blocks of the RS and its communication with MM and the end device:

Web services with MM: MM transfers the context information to the RS to establish a new connection between the RS and the end device.

Internal manager: This block creates and manages the streams in the RS to answer different users. This block also informs the MM about the changes on the related sessions.

3D Media & Render: According to the context information this block generates the initial objects needs for the composition and it can be modified in any moment by the user. It is also possible to interact with each of the elements in the scene.

Streaming server: It encodes the content in real-time and establishes the streaming session with the final client. All the initial parameters, both 3D Media and the streaming session are configured depending on the context information reported by the MM: Supported streaming protocols, codecs, screen size, etc. of the device, communication bandwidth, etc.

Web sockets for user interaction: Once the streaming communication is established between RS and end device, TCP sockets are used to assure low-latency. The user can interact with the content in three different ways: changes on the composition (add new elements, delete them, move their position, resize them, etc.); modifications over each object (3D movements, texture changes, stop or rewind a video or audio, etc.); and adjustments of the streaming parameters (video resolution, bitrate, codec, etc.).

C. END DEVICE

The capabilities required by *3DMaaS System* for the application of the end client are really affordable for most of the common home network devices. It only has to include a video streaming client showing the stream provided by the RS and scripting capabilities to send HTTP interaction 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.

IV. VALIDATION EXPERIMENTS

The *3DMaaS System* provides a wide range of video streaming formats in order to fit in very different devices. For example, the API to develop Yahoo! Connected TV applications available on Sony, Toshiba, Vizio or Samsung TV sets, supports RTSP protocol as well as Samsung Flash Applications do with RTMP. Other TV frameworks and set-top boxes implement HTTP live streaming e.g., Yahoo! Connected TV and Samsung Smart TV framework play HLS and other boxes as the Engel EN2000 and Humax iCord HD+ run HTTP chunked transfer encoding. However, each one brings feature singularities in terms of latency that must be considered when an interactive video application is designed to obtain a Quality of Experience comparable to applications running locally. Here, we tackle the tests followed by details about the implementation to measure the performance achieved with each streaming format.

The complete testing setup includes a RS server and PC clients to easily support the wide streaming standards spectrum. These end devices supply best scenario for low-latency without buffering through full capabilities. For this study, the server runs on top of a GNU/Linux Debian 6.0 Intel Core i7 3GHz and 6GB RAM, while the clients are represented by GNU/Linux Debian 6.0 Intel Core2 Quad 2.5Ghz and 4GB RAM devices. The elements involved are hosted at a local network in order to provide optimum network access.

We set up a RS of a *3DMaaS* remote rendering system with a server sending streams deployed over the open source multimedia framework Gstreamer[7] assuring a ultra low latency thanks to x264/MPEG-TS tune options[8]. The real time generated videos for each client are live streamed with a 320x240 resolution and 500 kbps and 1 Mbps bitrate per stream. It includes a simple scene with an OpenGL 3D cube enabling texture modification through client interaction performed by a bot that simulates random user interactivity to manipulate 3D scene. The client activity is time-traced locally and communicated to the server through a web socket. The time elapsed between the trace until the modification is perceptible by the client, this means remotely rendered, streamed and received in the client, is defined as the latency of the system.

The time elapsed between two consecutive frames establishes the minimum accuracy in terms of perceptible changes, so, in this case, the video framerate must be set up to avoid masking latency with low fps rates.

In order to define the limitations for live streaming of the standards, and the threshold for request scalability of the open source technological solutions that support our approach, we

[7] http://gstreamer.freedesktop.org/
[8] http://mewiki.project357.com/wiki/X264_Settings#tune

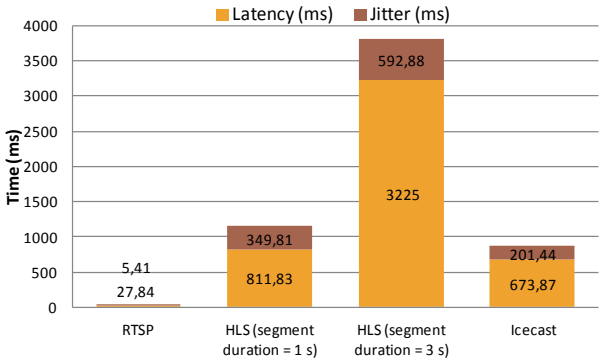

Fig. 3. Latency and Jitter over streaming protocol

have tested all the scenarios ranging over a number of concurrent clients, where the server provides a unique URL to each client.

Using the above mentioned setup, this analysis is based on measurements of the following scenarios:

A. RTSP

The RTSP server is built over a package provided by Gstreamer and due to the low latency score obtained in this case, the video is streamed at 120fps getting accurate measures.

B. HLS

Since HLS streaming is built on HTTP infrastructure elements, a standard Apache HTTP server (Apache/2.2.20, 64-bit) publishes media segments and grants HTTP access to the manifest file and consecutive updates. Often the recommended configuration for HLS stream includes 3 seconds segments and buffering of 3 segments which means high latency. However, the implementation of the Gstreamer plugins[9] achieves stable results with at least 1 second per fragment establishing segment duration-driven live streaming constraints. Moreover, smaller segments mitigate drawbacks if channel errors arise because it takes less time to download a new one and bitrate adaptation can occur sooner. According to the assessed delay the video is streamed at 60fps.

C. MPEG-DASH

Concerning this promising standard, GPAC[10] and Gstreamer solutions[11] provide the pillars of our system. However, it is important to highlight that due to sharing the HLS strategy, MPEG-DASH brings the same hurdles for live streaming.

D. Icecast

The Icecast2 server and the Gstreamer plugin for HTTP chunked transfer encoding of Ogg/Theora media provide underlying infrastructure to face this scenario. In this case the video is streamed at 60fps.

After running the experiments for the different video streaming formats, a comparison of collected latency scores reflects distant interactive performance. The tests carried out

[9] http://gitorious.org/ylatuya-gstreamer/gst-plugins-bad/commits/hlswip
[10] http://gpac.wp.mines-telecom.fr/2012/02/01/dash-support/
[11] https://github.com/ylatuya

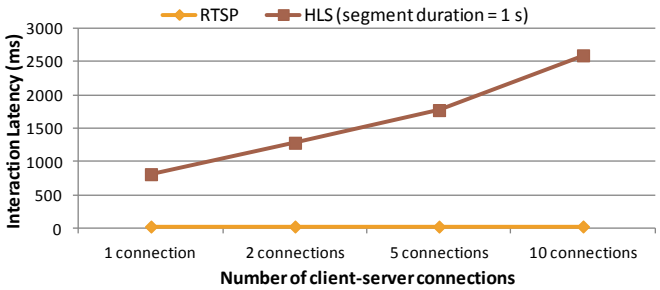

Fig. 4. Latency over number of concurrent streaming sessions

involve streaming about 500 minutes of video from the media server to client devices.

V. Results and Conclusions

The results of each of the test scenarios are plotted in Fig. 3. Here, RTSP rises as the most suitable protocol for low latency applications such as real time remote rendering and live streaming, achieving 27.84 ms score. While regarding the pull solutions Icecast and HLS for small segments obtain a similar performance. Another aspect is the interaction delay that is closely defined by the segment duration because the server has to package a video fragment containing the configured duration before being transferred to the client. A delay equivalent to the segment duration is introduced by this mechanism. Hence, the increase of this period has a negative impact on the interaction latency through all the segment driven technologies, including HLS and MPEG-DASH that reach a similar performance.

Making the segment duration longer reduces the number of fragments and provides a better compression because of the temporal redundancy. However, it also reduces the dynamic adaptability in terms of fragment-switching decisions.

Despite segment duration and pre-buffering recommendations usually do not tackle with real time scenarios, the specifications conceive real time streaming by decreasing segment duration even if the current implementations are not able to produce smaller segments.

The performance comparison achieved for concurrent access of multiple clients is depicted in Fig. 4. The RTSP server based on a standard package of Gstreamer deals with 50 concurrent unicast connections keeping latency features and performance while others are dramatic affected by additional workload mainly related to server-side Apache performance and concurrent disk accessing bottlenecks for storage of media segments.

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. Hence, the main reason for using HTTP rather than a dedicated streaming protocol such as RTSP/RTMP lays on standard HTTP infrastructure such as servers, caches, and CDNs can be used, avoiding problems with firewalls and NATs. However, segment-driven

mechanisms based on its duration have significant influence on the network performance of an HTTP streaming system. Despite the arguments for smaller segment sizes to allow dynamic adaptation to changeable networks and keep interaction latency, the majority of implementations put aside delay feature focusing on buffered live broadcasting applications.

This paper focuses on assessing the performance of a wide range of emerging streaming standards. The investigated implementation is built on top of ongoing open source plugin projects for Gstreamer. In the experimental results the codec x264 combined with Gstreamer RTSP server show the potential for managing a high demanding scenario with multiple concurrent users watching a customized video stream.

All these results demonstrate that mature RTSP solutions based on UDP and RTP protocols are still more suitable standards than interoperable new ones in order to deliver real time media with live stream applications, especially to support advanced 3D experiences in ubiquitous scenarios.

The future work will focus on implementing an adaptive streaming solution using HLS and MPEG-DASH addressing the latency drawback. This would include aggregating mechanisms to bring closer HTTP to RTSP behaviour by means of techniques to reduce the segment duration up to the RTP/UDP packets ones to the detriment of encoding efficiency. This means the common practice to start a segment with an intra coded frame (I-Frame) to provide random access and independent decoding would be over exploited by minimizing both the segment size and the distance between I-Frames, which also boost the dynamic context adaptation of the system.

3DMaaS System provides a feasible solution to address interoperable delivery of advanced 3D Media applications to home network thin devices which can enjoy rich experiences that need of specific high performance hardware. Moreover, *3DMaaS* deals with heterogeneous application latency requirements and generates different streaming formats to fit with the specific device capabilities that tend to be aligned with HTML5 and HbbTV standards.

REFERENCES

[1] Zahariadis, T., Daras, P., Laso, I.: “Towards Future 3D Media Internet”. In: NEM Summit 2008. Saint-Malo, France, October 13-15 (2008)

[2] P. Daras and F. Alvarez: "A Future Perspective on the 3D Media Internet”, Towards the Future Internet - A European Research Perspective, IOS Press, May 2009

[3] Anttonen, M.; Salminen, A; Mikkonen, T; Taivalsaari , A: “Transforming the web into a real application platform: New technologies, emerging trends and missing pieces”. Proceedings 2011 ACM Symposium on Applied Computing pp. 800-807 (2011)

[4] Ortiz, S.: “Is 3d finnally ready for the web?” Computer (1), 14-16 (2010).

[5] Begen, A.; Akgul, T.; Baugher, M.; "Watching Video over the Web: Part 1: Streaming Protocols," Internet Computing, IEEE , vol.15, no.2, pp.54-63, March-April 2011.

[6] Begen, A.; Akgul, T.; Baugher, M.; "Watching Video over the Web: Part 2: Applications, Standardization, and Open Issues," Internet Computing, IEEE, vol.15, no.3, pp.59-63, May-June 2011.

[7] Rubio L; “A Dynamic Adaptive HTTP Streaming Video Service for Google Android”. PhD. dissertation, School of Information and Communication Technology (ICT), Royal Institute of Technology (KHT). Stockholm, Sweden. October 2011.

[8] Hofmann, I.; Farber, N.; Fuchs, H.; "A study of network performance with application to Adaptive HTTP Streaming", Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium pp.1-6, 8-10 June 2011.

[9] Sodagar, I.; "The MPEG-DASH Standard for Multimedia Streaming Over the Internet", Multimedia, IEEE, vol.18, no.4, pp.62-67, April 2011.

[10] Lohmar, T.; Einarsson, T.; Frojdh, P.; Gabin, F.; Kampmann, M.; "Dynamic adaptive HTTP streaming of live content", World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium pp.1-8, 20-24 June 2011.

[11] Kalker, T.; Samtani, R.; Xin Wang; "UltraViolet: Redefining the Movie Industry?" Multimedia, IEEE, vol.19, no.1, pp.7, Jan. 2012.

[12] Zorrilla M., C.M.U.A.M.J.F.O.D..O.I.G.: "Next Generation Multimedia on Mobile Devices." Mobile Technology Consumption: Opportunities and Challenges. IGI Global (2012).

M. Zorrilla is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net. He received his Telecommunication Engineering degree in 2007 from Mondragon Unibertsitatea, Spain.

He focuses on the research lines related to multimedia services and interactive media technologies. During his studies he worked in the area of communications of IK4 Ikerlan Research Centre (www.ikerlan.es) (2002-2006). Afterwards, he developed there the End of Degree Project about The Transport of Multimedia Traffic With an “Industrial Ethernet” Communication Bus (2006-2007). Since 2007 he is working at Vicomtech-IK4, where he designs, develops and leads projects of the Digital Television and Multimedia Services area.

A. Martin is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net. He received his engineering degree in 2003 from University Carlos III, Spain.

He starts his Phd in the Communications and Signal Processing Department of the University Carlos III in the video coding research area in 2003. At the same time he collaborates with Prodys developing a standard MPEG-4 AVC/H.264 codec for DSP. In 2005 he starts to work on Telefonica I+D in projects related to multimodal interactive services for Home Networks. Also in Telefonica I+D, he goes deeper into image processing area in terms of 3D video and multiview coding. From 2008 to 2010 he worked in Innovalia as R&D Project consultant related with smart environments and ubiquitous and pervasive computing. Currently he is on Vicomtech-IK4 managing and developing R&D projects around multimedia content services.

F. Mogollon is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net. He received his Telecomunication Engineering degree in 2006 from Universidad de Cantabria, Spain. His research interests are in the areas of multimedia services and interactive media technologies. He had been a former developer at Zitralia Security Solutions (2006-2008). Since 2008 July he is working at Vicomtech-IK4, where he designs and develops projects.

J. Garcia is with the Department of Digital Tv & Multimedia Services, Vicomtech-IK4 GraphicsMedia.net. He received his Telecommunication Engineering degree in 2008 from Mondragon Unibertsitatea, Spain. His research interests are in 3D and real-time professional TV broadcast productions. Since 2008 he is working at Vicomtech-IK4, where he designs and develops projects.

I. Olaizola is the head of Digital TV and Multimedia Services Department in Vicomtech-IK4 GrpahicsMedia.net, Spain. He received his Engineering degree from University of Navarra, Spain (2001).

He developed his Master thesis at Fraunhofer Institut für Integrierte Schaltungen (IIS), Erlangen -Germany- 2001 and currently he is preparing his PhD in Computing Science and Artificial Intelligence at University of Basque Country. He has participated in many industrial projects related with Digital TV as well as several European research projects in the area of audiovisual content management. His current research interests include multimedia content analysis frameworks and techniques to decrease the semantic gap.