Semantic multimedia remote display for mobile thin clients

B. Joveski, M. Mitrea, P. Simoens, I.J. Marshall, F. Prêteux, B. Dhoedt

Abstract

Current remote display technologies for mobile thin clients convert practically all types of graphical content into sequences of images rendered by the client. Consequently, important information concerning the content semantics is lost. The present paper goes beyond this bottleneck by developing a semantic multimedia remote display. The principle consists of representing the graphical content as a real-time interactive multimedia scene-graph. The underlying architecture features novel components for scene-graph creation and management, as well as for user interactivity handling. The experimental setup considers the Linux X windows system and BiFS/LASeR multimedia scene technologies on the server and client sides, respectively. The implemented solution was benchmarked against currently deployed solutions (VNC and Microsoft-RDP), by considering text-editing and www-browsing applications. The quantitative assessments demonstrate: (1) visual quality expressed by seven objective metrics, *e.g.* PSNR values between 30 and 42dB or SSIM values larger than 0.9999; (2) downlink bandwidth gain factors ranging from 2 to 60; (3) real-time user event management expressed by network roundtrip-time reduction by factors of 4 to 6 and by up-link bandwidth gain factors from 3 to 10; (4) feasible CPU activity, larger than in the RDP case but reduced by a factor of 1.5 with respect to the VNC-HEXTILE.

Index Terms—semantic multimedia remote display, mobile thin client, MPEG-4 multimedia scene (BiFS, LASeR), X Window System, VNC-HEXTILE, RDP.

1. Introduction

In accordance with current day user expectancies, no functional discrepancies should be noticeable between mobile thin client and fixed desktop applications. Under this framework, the definition of a multimedia remote display for mobile thin clients remains a challenging research topic, requiring at the same time a high performance algorithm for the compression of heterogeneous content (text, graphics, image, video, 3D, …) and versatile, user-friendly real time interaction support [1], [2]. The underlying technical constraints are connected to the network (arbitrarily changing bandwidth conditions, transport

B. Joveski and M.Mitrea are with the ARTEMIS Department at Institut Telecom, Telecom SudParis, 9, Rue Charles Fourier, 91011 Evry Cedex, France. E-mail: {bojan.joveski, mihai.mitrea} @telecom-sudparis.eu.
P. Simoens, and B. Dhoedt are with the Department of Information Technology at Ghent University-IBBT, Gaston Crommenlaan 8 bus 201, B-9050 Gent Belgium. E-mail: {pieter.simoens, bart.dhoedt} @intec.ugent.be.
P. Simoens is also affiliated with the Dept. INWE, Ghent University College, Valentyn Vaerwyckweg 1, B-9000 Gent, Belgium.
I.J. Marshall is with the Department of Research at Prologue, 12 Avenue Tropiques 91940 Ulis (Les), France. E-mail: ijmarshall@proloque.fr.
F. Prêteux is with the Research Department of MINES ParisTech, 60 Boulevard Saint-Michel 75272 PARIS Cedex 06, France. E-mail: francoise.preteux@mines-paristech.fr

errors, and latency), to the terminal (limitations in CPU[1], storage, and I/O resources), and to market acceptance (backward compatibility with legacy applications, ISO compliance, terminal independence, and open source support).

In order to develop remote display applications for wired environments, several reference technologies are available: X [3], VNC [4], NX [5], RDP [6], to mention but a few. Regardless of its original type, the heterogeneous graphical content (text, image, graphics, video, 3D, …) is converted into sequences of images (eventually a mixture of images and graphics), which are then interactively displayed on the terminal. Such an approach would appear to be inappropriate when addressing the above-mentioned mobile thin client constraints [7]-[10] since it would reduce the user experience. Moreover, these solutions are restricted practically by their genericity: they depend on the device capabilities, operating system and user community support. This is a consequence of the large variety of both hardware and software mobile thin clients on the market [11] and represents a pitfall for a standard deployment. From the software point of view, Android, iOS and RIM are the leading operating systems in US (with 39%, 28% and 20%, respectively), while the penetration of the Windows mobile/WP7 does not exceed 9%. From the hardware point of view, two different strategies are followed. While Apple, Blackberry and HP consider only one operating systems for their smarphones (iOS, RIM and Palm OS, respectively), other manufacturers address multiple OS. For instance, the Samsung and HTC offer a choice between Android and Windows mobile/WP7.

The challenge of exploiting virtualized screen technologies for ensuring cloud-mobile convergence is taken up in [12] where an image-oriented architecture is advanced. Although the integration of such an architecture into the www browsing use cases is illustrated, no objective evaluation of the related performances (bandwidth and CPU consumption) is reported.

This present paper advances a semantic multimedia remote viewer. The principle consists of designing an MPEG-4 (BiFS/LASeR) software architecture that transparently ensures the bidirectional exchange of multimedia content between a server and a user terminal. This architecture is centered on the concept of an interactive multimedia scene-graph and features novel components for its creation and management, as well as for user interactivity handling.

The paper has the following structure. Section II provides the main definitions as well as a critical analysis of the state-of-the art relating to remote display technological and applicative supports. Section III highlights the existing scene-graph representation technologies. Section IV presents a semantic multimedia MPEG-4 (BiFS/LASeR) based architecture for a mobile thin client remote display. The benchmarking of this architecture against currently deployed solutions (VNC-HEXTILE and Microsoft RDP) considers text editing and www browsing applications and is described in Section V. Section VI discusses the potential acceptance of the present solution by the industrial world while Section VII concludes the paper and open perspectives for future research work.

[1] All acronyms used throughout the paper are listed in the List of Abbreviations, to be found at the end of the paper.

2. State Of the Art

2.1. Definitions

In the widest sense, the *thin client* paradigm refers to a terminal (desktop, PDA, smartphone, tablet) essentially limited to I/O devices (display, user pointer, keyboard), with all related computing and storage resources located on a remote server farm. This model implicitly assumes the availability of a network connection (be it wired or wireless) between the terminal and the computing resources.

From the architectural point of view, the thin client paradigm can be accommodated by a classical client-server model, where the client is connected to the server through a connection managed by a given protocol, Figure 2. From the functional point of view, the application (text editing, www browsing, multimedia entertainment, …) runs on the server and outputs semantically structured graphical output (*i.e.* a collection of structured text, image/video, 2D/3D graphics, …). This graphical content is generally converted into sequences of images, subsequently transmitted and visualized on the client, where the user interactivity is captured and sent back to the server for processing.

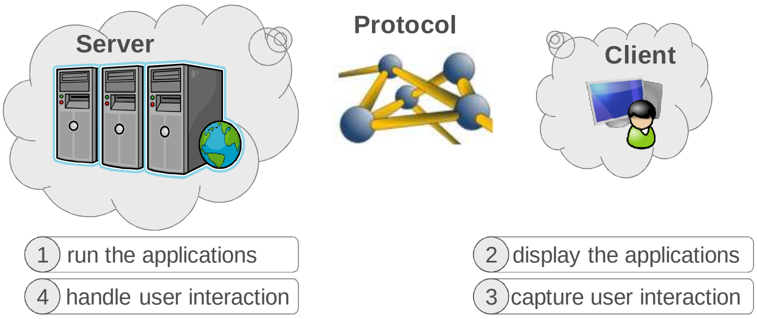

Server
Protocol
Client
1 run the applications
4 handle user interaction
2 display the applications
3 capture user interaction

Figure 2. Remote display framework.

Within the scope of this paper, the term *remote display* refers to all the software modules, located at both end points (server and client), making possible, in real time, for the graphical content generated by server to be displayed on the client end point and for subsequent user events to be sent back to the server. When these transmission and display processes consider, for the graphical content, some complementary semantic information (such as its type, format, spatio-temporal relations or usage conditions, to mention but a few) the remote display then becomes a *semantic remote display*[2].

Our study brings to light the potential of *multimedia scene-graphs* for supporting semantic remote displays. The concept of the scene-graph emerged with the advent of the modern multimedia industry, as an attempt to bridge the realms of structural data representation and multimedia objects. While its definition remains quite fuzzy and application dependent, in the sequel we shall consider that a scene-graph is [13]: "*a hierarchical representation of audio, video and graphical objects, each represented by a […] node abstracting the interface to those objects. This allows manipulation of an object's properties, inde-*

[2] The usage of the word *semantic* in this definition follows the MPEG-4 standard specification [13] and the principles in some related studies [14], [15].

pendent of the object media." Current day multimedia technologies also provide the possibility of direct interaction with individual nodes according to user actions; such a scene graph will be further referred to as an *interactive multimedia scene-graph*.

2.2. Off-the-shelf technological support

These days, all remote display solutions (be they wired or wireless, desktop computer or thin client oriented, Windows or Unix based, *etc.*) exploit the client-server architecture. Consequently, any remote display technological support can be assessed according to the following three criteria: (1) the interception of the visual content generated by the application at the server side, (2) the compression and the transmission of the content to the client, and (3) the management of the user interactivity (including the transmission of user events from client to server). An additional fourth criterion related to the energy consumption is taken into consideration for mobile thin clients. The study in [16] brought to light that the energy consumption of a smartphone depends on the network (GSM/Wi-Fi), the CPU, the RAM, the display and the audio. While the last three factors are rather more related to the device and to the actual user behavior, the amount of data transmitted through the network and the CPU activity intrinsically depend on the technology and will be further investigated in our study.

The present section considers the most frequently encountered remote desktop viewers support technologies (X window, NX, VNC, and RDP) for discussion according to these four criteria.

The **X window system** represents the native remote viewer for all current day desktop Linux applications. The application output, intercepted by an XClient (running on the server[3]), is represented by a rich set of 128 basic graphic primitives [3] describing the X11 protocol. Such X content is transmitted to the XServer (running on the client) by the X11 protocol, which makes no provision for content compression. On the client side, the XServer not only displays the graphical content but also captures the user interactivity by generic Linux OS mechanisms (keyboard/mouse drivers). While ensuring good performance in wired desktop environments, the X window system cannot be directly employed for mobile thin clients, mainly because of its bandwidth and latency requirements; actually, no X window system for thin clients is currently available.

By providing alternative protocols, the **NX technology** (a proprietary NoMachine solution) is intended to reduce the X11 network consumption and latency. Assuming an X window system is already running locally (the X Client, the X11 protocol and the X Server being all accommodated by the server), an NX proxy intercepts the X11 protocol, compresses it and subsequently transmits the result to the NX agent running on the remote client. Note that no particular user interactivity mechanisms have been developed. Although the experiments showed a good compression rate of the initial X11 content, such a solution is not yet available.

[3] The X window system terminology may be confusing, the user's terminal being the XServer and the server application being the XClient [3].

The **VNC** (**Virtual Network Computing**) remote viewer also assumes that an X window system is already available locally on the server side and brings new components in order to alleviate bandwidth and CPU constraints. The VNC Server (running on the server) intercepts the X graphical content at the XServer side. It further converts it to raw images (pixel maps) which are subsequently compressed using image compression algorithms and transmitted to the client by using the RFB (Remote FrameBuffer) protocol. On the VNC Client side (running on the client), the visual content composed only of images can be directly displayed. As in the previously discussed cases, the user interactivity is managed by the underling operating system. Several image compression optimizations are currently considered by VNC: TightVNC, TurboVNC, VNC-HEXTILE, VNC-ZRLE, … . For mobile thin clients, VNC-HEXTILE represents nowadays the most effective solution of that kind. Despite completely disregarding semantic information concerning the visual content to be displayed, VNC may be considered today as the most intensively used mobile thin client remote viewer: its myriads of versions for Linux, Andorid, iOS, RIM and even Windows Mobile can cover more than 87% of the personal smartphones in US [11].

Microsoft Windows OS provides the **RDP (Remote Display Protocol) framework**, a proprietary client-server solution for remote displays, available in both desktop and mobile versions. On the server side, the RDP server intercepts the application output through the GDI (Graphical Device Interface) and represents it by a mixture of images, graphics and formatted text. This content is then transmitted using the RDP protocol to the RDP client where it is displayed. In desktop environments, the RemoteFX, an emerging extension of the basic RDP framework, also enables multimedia content transmission [17]. The user interactivity is managed by RDP and/or Windows OS drivers. Although natively designed for the Windows OS, Linux-based RDP servers also emerged in the last months. Nowadays, the RDP clients target about 9% of the US smartphone market [11] and it is forecasted to have the most important relative growth by 2015 [18].

These four basic technologies are combined in practice into a myriad of thin client solutions, as explained in Appendix 1.

2.3. Discussions

The performances exhibited by the remote display technologies presented in Section 2.1 are synoptically illustrated in Figure 3. It can be noticed that these technologies feature no direct support for multimedia (except for the RDP RemoteFX in desktop environments), none of them are compatible with the ISO multimedia standards and several requirements are still to be met when designing a mobile thin client remote display:

- interception of visual content: capturing the graphical content at the lowest possible levels (thus ensuring generality and good visual quality) while retaining the semantic information of the content (thus preserving the content type and providing multimedia experience);

- visual content compression/transmission: deploying an efficient compression algorithm for the handling of heterogeneous content;
- user interactivity: ensuring a prescribed QoE (Quality of Experience) for the user interactivity, irrespective of the application (text editing, www browsing, entertainment, …), of the network bandwidth (both up-link and down-link) and of the type of terminal;
- CPU activity: specifying low-complexity algorithms, coping with the CPU limits imposed by the thin clients.

The present paper reports on the possibility of using the MPEG-4 technologies for multimedia scenes to jointly solve these four issues (see the *Targeted solution* in the low-left area in Figure 3).

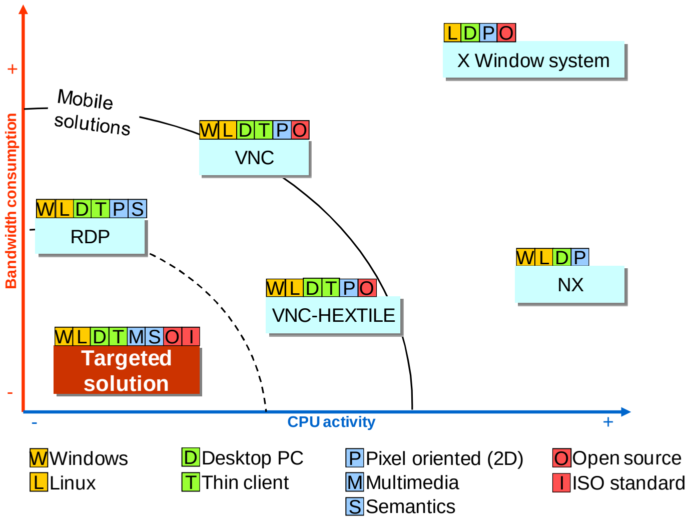


Figure 3. Comparison of the current remote display solutions.

3. Multimedia Scene-graph Representations

Under the framework of MPEG-4 standard, a dedicated multimedia scene description language, called Binary Format for Scene (BiFS) [19], [20] is defined. It describes the heterogeneous content of the scene, manages the scene object behavior (*e.g.* object animation), ensures the timed and conditional updates (*e.g.* user input/interactivity) and encodes each object by its own coding scheme (video is coded as video, text as text, and graphics as graphics).

The BiFS principles have been further optimized for thin clients and mobile network purposes, thus resulting in a standard called Lightweight Application Scene Representation (LASeR) [21], [22].

The existing technologies for heterogeneous content representation (BiFS, LASeR, Adobe Flash [23], Java [24], SMIL/SVG [25], TimedText [26], xHTML [27], see Figure 4) can be benchmarked according to their performances in the areas of binary compression, dynamic updates, streaming and user interactivity management.

Binary compression for multimedia scenes is already offered by several solutions, both on the inside (BiFS and LASeR) and on the outside (Flash and Java) of the MPEG world. On the one hand, LASeR is the only technology specifically developed addressing the needs of mobile thin devices requiring at the same time strong compression and low complexity of decoding. On the other hand, BiFS takes the lead over LASeR by its power of expression and its strong graphics features which can describe 3D scenes. A particular case is represented by the xHTML technology which has no dedicated compression mechanism, but exploits some generic lossless compression algorithms (*e.g.* gzip) [28], [29].

Dynamic updates allow the server to modify the multimedia scene in a reactive, smooth and continuous way [30]. In this respect, commands permitting scene modifications (object deletion / creation / replacement) in a timely manner [30] should be provided inside the considered technology. This is the case of BiFS, LASeR and Flash. xHTML does not directly allow dynamic updates, but delegates this responsibility to additional technology (*e.g.* JavaScript).

Streaming refers to the concept of consistently transmitting and presenting media to an end user at a rate determined by the media updating mechanism *per se*; *live streaming* refers to the instantaneous delivery of some media created by a live source. BiFS and LASeR are the only binary compressed content representations intrinsically designed to be streamed. In this respect, dedicated mechanisms for individual media encapsulation into a binary format have been standardized and generic transmission protocols are subsequently employed for the corresponding streams. Note that the Flash philosophy does not directly support such a distribution mode: the *swf* file is generated on the server and then downloaded to the client which cannot change its functionalities. However, inside the *swf* file, Flash does provide tools for streaming external multimedia contents with their own native support, *e.g.* a FLV video can be streamed inside the Flash player. A similar approach is followed by xHTML.

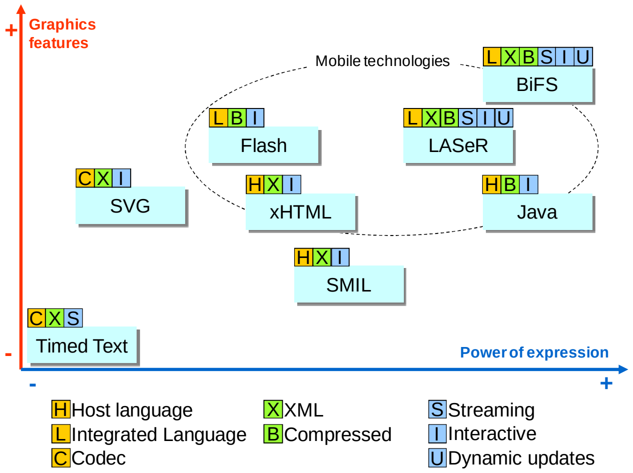


Figure 4. Concurrent solutions for heterogeneous content compression, updating and streaming (this figure was obtained by extending a similar representation in [21]). Power of expression: possibility of describing complex/heterogeneous scenes. Graphics features: visual quality of the displayed content.

Nearly all the technologies in Figure 4 are concerned with two well-defined ways for handling user interactivity captured at the scene level: client-side and server-side.

On the one hand, client-side interactivity deals with content manipulation on the end user terminal, where only local scene updates are available: the user events are captured and the scene is correspondingly updated, without contacting the server.

On the other hand, server-side interactivity supposes that the user events are sent to the server by using an up-link channel. MPEG-4 provides two possible solutions for ensuring the server-side interactivity. First, the ECMA script (JavaScript language) [31] can be considered in order to enable programmatic access to MPEG-4 objects. In order to achieve server-side interactivity, an AJAX HttpRequest [32] object is used to send user interactivity information to the server. Such a solution is not only common to BiFS and LASeR, but also to Flash and xHTML. In the particular case of BiFS, a second interactivity mechanism is provided by the *ServerCommand* which allows the occurrence of a user event to be directly signaled from the scene to the server.

To conclude with, MPEG-4 BiFS and LASeR are potentially capable of fulfilling all four remote display challenges (Section 2.4):

- the heterogeneous content generated by the application can be aggregated into a multimedia MPEG-4 scene-graph, and the related semantic information can be used for the management of this graph;
- the compression of each type of content (text, audio, image, graphics, video, 3D) by dedicated codecs and the related live streaming are possible by using the corresponding BiFS/LASeR technologies;
- the user interactivity can be established both locally and remotely;
- the client CPU activity may concern only light-weight operations (scene graph rendering and basic user event handling) while the computational intensive operations (scene graph creation/management and user event management) may be performed by the server.

In addition to these technical properties, BiFS and LASeR also present the advantage of being stable, open international standards, reinforced by open source reference software.

In their previous works, the authors already disclosed the basic idea of designing a multimedia remote viewer by converting a limited set of the X graphic primitives into BiFS [33] and LASeR [34]. The present paper goes one step further, by presenting a comprehensive architecture (Section IV) and by validating the underlying prototype for text editing and www browsing applications (Section V).

4. Developed Architecture

As mentioned above, traditional remote display solutions are based on the conversion of the original content into sequences of images, Figure 5. These images are subsequently compressed, transmitted and rendered according to image/video principles and tools. Each remote viewer application comes with its own means for capturing the user interaction at the level of the OS drivers. The present section goes beyond the image limitations and advances a semantic mobile thin client remote display architecture, centered on the MPEG-4 interactive multimedia scene technologies, Figure 6. In order to benefit practically from such technologies, a scene Scene-graph Management Module is designed and implemented. The content is then compressed and transmitted, according to open-standard/open-source tools. On the client side, the user events (key strokes, mouse clicks, *etc*.) are captured in an ISO standardized manner and are subsequently managed by an architectural block devoted to this purpose.

An overview will be presented in Section 4.1, followed by implementation details in Section 4.2.

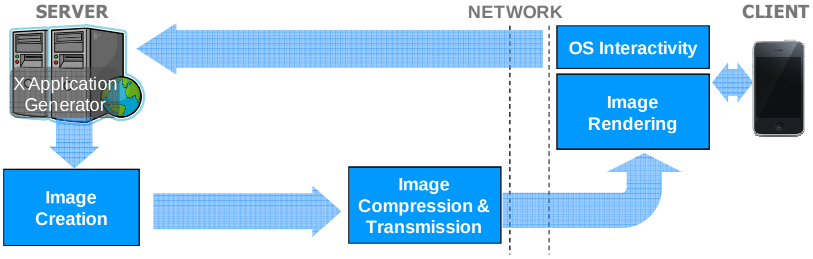


Figure 5. State-of-the-art architectural framework for mobile thin client remote display.

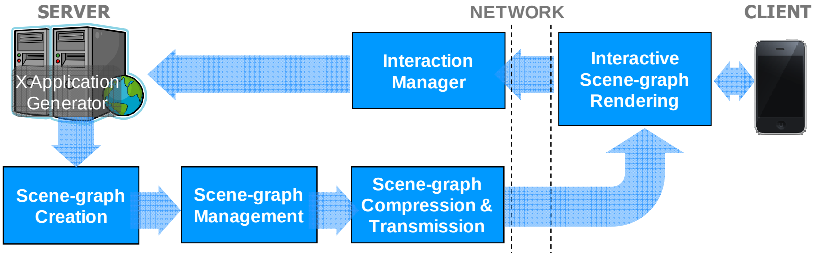


Figure 6. Advanced architectural framework for mobile thin client remote display.

4.1. Architectural synopsis

The application generator creates X11 graphical content that is to be presented at the client; it corresponds to the traditional application (be it text editing, www browsing, …) which is kept unchanged (*i.e*, from the application point of view, our architecture is completely transparent). Figure 6 explicitly considers X applications running on Linux servers; however, the architecture is general and can be instantiated on any OS, such as Windows or Apple, for instance.

The Scene-graph Creation module performs three tasks. It detects the graphical primitives generated by the X application, parses them and subsequently translates them into a multimedia scene-graph preserving not only the multimedia content but also its semantic. The Scene-graph Creation module was designed so as to meet the first requirement set in Section 2.3*:* the content generated by any X legacy application can be represented by a semantic multimedia scene-graph, without changing that application. The underlying technical challenges are related to the completeness (*i.e.* the possibility of converting all the visually relevant X primitives) and flexibility (*i.e.* the possibility to integrate future X extensions with minimal impact in the architecture). To our best knowledge, no work on that subject has already been reported.

The Scene-graph Management module ensures the dynamic, semantic and interactive behavior of the multimedia scene-graph. In this respect, the previously created scene-graph is enriched with logical information concerning its content type, its semantic and its related time of evolution as well as with user interactivity. The Scene-graph manager addresses the second and the fourth requirements in Section 2.3: provide a heterogeneous content which can be subsequently optimally compressed (the optimality refers here not only to the trade-off of visual quality-bandwidth but also to the CPU activity). The innovation is related to the specification of an algorithm enabling the dynamical updating of the scene-graph according to the network/client/server conditions (be them real-time or evaluated on a short history).

The Compression & Transmission module is in charge of the creation of a binary encoded stream (the compressed scene graph) which is subsequently streamed live to the client. The technical challenge is related to the flexibility of the transmission mode (unicast, broadcast, multicast) and of the transmission protocol.

The Interactivity Manager maps the user events back to the application, thus ensuring the server-side interactivity and contributing to the scene-graph management process. This module is designed according to the third requirement set in Section 2.3 (related to the user interaction). As in the Scene-graph creation module case, its technical challenges are related to the completeness of the solution and to its flexibility.

The Interactive Scene-graph Rendering is ensured by a multimedia player which also captures the user interactivity and lets the local interactive scene graph handle it. This module is designed according to the third and four requirements set in Section 2.3: the user interaction is captured in a standard way, at the scene-graph level while the rendering process demands in terms of CPU activity should not exceed the objective limits set by the nowadays thin clients.

The *Network* ensures the traffic from the server to the thin client (the streaming of the interactive scene-graph) and *vice-versa* (information concerning the user interaction).

4.2. Implementation details

This architectural framework is instantiated on a Linux virtual machine (VM) as a server and on a smart phone as a thin client, Figure 7. The actual implementation considers a server based on the Ubuntu distribution accommodating the server components and a Windows mobile thin client accommodating an MPEG-4 player. The network is established using a wireless protocol (the actual implementation considered a Wi-Fi 802.11g network).

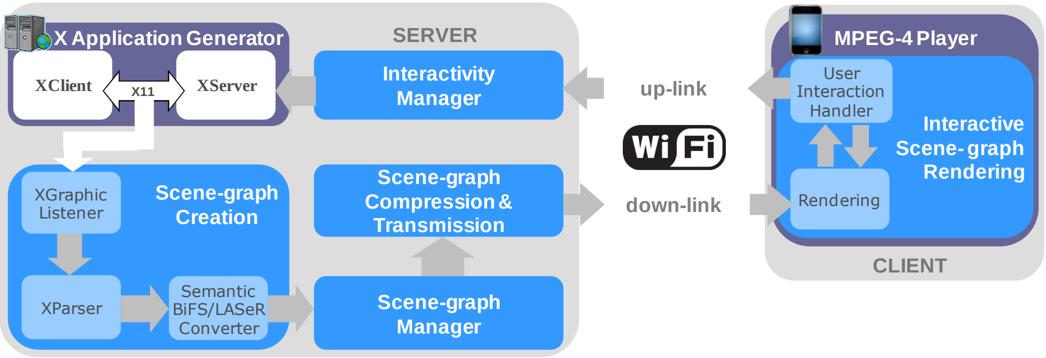


Figure 7. Detailed architectural framework.

4.2.1. Server-side components

The X Application Generator is implemented by an X window system (**XServer**, **XClient** and **X11**) exploited by a traditional application (text editing, www browsing, …). In order to cope with the backward compatibility constraint, the X window system is kept unchanged during the present study.

The Scene-graph Creation is implemented by three blocks, each performing one of the previously described tasks: the XGraphic Listener, the XParser and the Semantic BiFS/LASeR convertor. The need for such a module as well as its restricted functionality version were introduced in [33], [34]; the present paper presents an integrated module, covering all the X basic primitives, lively generated by the X application.

XGraphic Listener: Located between the XClient and the XServer, by listening on a socket through which they communicate using the XProtocol, the graphic listener intercepts the X11 messages and passes the results to the XParser. By developing the XGraphical Listener as an independent architectural component, completely transparent to both XClient and XServer, the backward compatibility and the Unix-based OS independence are jointly ensured (no application modification or driver development is required). Moreover, no functional limitation is induced by listening to the XProtocol instead of intercepting the visual content directly at the XClient side: all graphical information is available at the protocol level and no network overcharge is produced (the XClient and the XServer run locally). Moreover, semantic information related to that graphical content is also available at this level; consequently, the architecture presented in Figures 6 and 7 require no sophisticated segmentation/tracking/scheduling algorithms.

XParser: This component was developed for the parsing of the XProtocol in order to extract the graphical primitives and their related semantics to be presented to the Semantic BiFS/LASeR converter. Just as an illustration, consider the following case in which we are interested in the *PolyRectangle* request; its complete X syntax is the following:

```
1 bytes        67 opcode                     // the request message ID
1 bytes        unused
2 bytes        3+2n requestlength            // the length of the request message
4 bytes        DRAWABLE drawable             // the parent of the graphic primitive
4 bytes        GCONTEXT gc                   // the description of the rectangle material
8n bytes       LISTofRECTANGLE rectangles    // list of rectangles described with position and size
```

In order to parse this message from the X11 protocol, the following code can be used:

```
drawable = x11application->getUInt32(&(message[0]));
graphicalContent = x11application->getUInt32(&(message[4]));
noRectangles = x11application->getUInt16(&(header[2] ))– 3 / 2;
for (i=0; i < noRectangles; i++)
{
        x=x11application->getUInt16(&message[8 + 8 * i ]);
        y=x11application->getUInt16(&message[8 + 8 * i + 2]);
        width=x11application->getUInt16(&message[8 + 8 * i + 4 ]);
        height=x11application->getUInt16(&message[8 + 8 * i + 6]);
}
```

Semantic BiFS/LASeR Converter: Each X request intercepted by the parser is mapped to a function which converts it to its BiFS/LASeR counterpart: all of the 128 basic X visual primitives (rectangle, line, circle, *etc*…), text and images [3] have already been successfully converted. Assuming the X window system will be extended in the future with other graphical primitives, this component should be able to evolve so as to cope with these updates. Although it is not possible today to foresee the syntax of these extensions, the possibility of converting them in BiFS/LASeR elements is guaranteed even when no straightforward MPEG counterparts would be available: in the worst case scenario, these future graphical elements would be rendered and the corresponding pixel maps would be included in the MPEG scene-graph.

Note that the Semantic BiFS/LASeR Convertor also allows semantic information about the X content to be converted for use in the management of the MPEG-4 scenes.

When considering the example above, the following BiFS conversion was obtained[4]:

[4] The result of the conversion is presented in MPEG-4 BiFS Textual format (BT); an equivalent and alternative way of representing uncompressed BiFS content would be the XMT-A (e**X**tensible **M**PEG-4 **T**extual) format, an XML-based representation defined in [13].

```
Transform2D {
     translate x y
     children [
          Shape {
               appearance Appearance {
                    material Material2D {
                    }
               }
               geometry Rectangle {
                    size width height
               }
          }
     ]
}
```

This corresponds to the following LASeR description (SVG format):

```
<rect width="" height="" x=”” y=”” style="fill:rgb(,,);stroke-width:1; stroke:rgb(,,)"/>
```

Note that in contrast to the BiFS situation, not all the X graphic primitives have a straightforward conversion in LASeR. For instance, LASeR makes no provision for describing raw images, which are generated by the XClient through the *PutImage* primitive. In such a case, more elaborated scene management mechanisms are provided. For instance, in order to convert the *PutImage* primitive, the related pixel buffer corresponding to a raw image is first converted into a png binary buffer. This buffer is then base64 encapsulated and mapped to the LASeR *Image* node.

Of course, in our study, BiFS and LASeR are not operating at the same time (they are alternatively enabled, in order to ensure a comparison of their performances).

Scene-graph Manager: As previous explained, this component ensures the dynamic, semantic and interactive behaviors of the MPEG-4 scene-graph. The study in [35] hinted to the idea that a supplementary logic layer can be added over the BiFS scene in order to reduce the bandwidth consumption; the present paper advances a comprehensive management module, exploiting the X graphical content semantics in order to improve the bandwidth/memory/complexity trade-off for MPEG-4 thin clients.

The dynamic and semantic evolution of the scene-graph can be managed by combining the information generated by application with the semantic tagging of the scene graphic elements and a prescribed set of logic rules concerning the possible re-usage of the most common graphic elements (*e.g.* menus, icons, …) and/or the adaptation of the content to the actual network client conditions. The current implementation is based on three main principles. First, by exploiting the semantic information about the elements composing the scene-graph, some *a priori* hints about their usage can be obtained. For instance, when typing, the most frequent letters/words represent the most frequent scene updates; hence, an important network bandwidth gain would be achieved when caching this content on the client side for its re-usage. This gain would be even more important when considering menus, icons or particular images during www browsing. Secondly, as the thin client has limited memory resources, a pruning mechanism, controlling the caching persistency is required. Finally, note that the mobile network conditions are likely to change significantly even during short periods of time. Such a situation can seriously impede the trade-off between the user experience and the bandwidth consumption: when the bandwidth drops, a high quality content would overcharge the traffic; conversely, when the bandwidth increases, a low quality content would frustrate the user. In order to address this issue, our solution considers real-time adaptation of the encoding parameters.

As it can be seen, the Scene-graph Manager has an inner methodological and technical complexity and its in-depth description is outside of the scope of the present paper. However, the Appendix 2 included in the present paper illustrates these three principles for the particular case of image re-use.

In order to ensure the user interactivity mechanisms, basic MPEG-4 elements, referred to as *sensors* [13], are considered in the multimedia scene-graph.

At the output from this block, interactive semantic multimedia content, ready to be streamed, is provided.

Compression & Transmission: This module integrates the GPAC libraries for the binary encoding of the BiFS/LASeR graphical content [36]-[37] and the streaming support from the LIVE555 Streaming Media [38]. The input to the streamer is BiFS/LASeR MPEG-4 stream content while its output is sent to the thin client by using RTSP/RTP. Note that nowadays the GPAC is the only open-source, publically available reference software framework for BiFS/LASeR; hence, its usage is implicitly compulsory. However, the use of LIVE555 and of RTSP/RTP was an implementation choice guided by their versatility (connection mode, usage of the protocol and streaming buffer control). According to the targeted application, the streaming tool can be changed, without affecting the rest of the architecture.

Interactivity Manager: It receives the user event, sent through the up-link (see Section 4.2.2 below), by the *Interactive Scene-graph Rendering* module. In the current implementation, all the keyboard and mouse/touch screen events are supported. The Interactivity Manager converts these events into the syntax required by the XServer which ensures the server side interactivity mechanisms, *i.e.* it updates the X Application (XServer updates).

For instance, a click event captured by the MPEG-4 sensors can be converted into the X syntax by the following code:

```
if(leftButton==0) {

  //getting the time of the day
  gettimeofday(&currentTime,NULL);

  //setting the last click moment
  lastClickTime = currentTime;

  //Posting the button event
  conv->PostButtonEvent(MT_BTN_LEFT,MTBUTTON_DOWN,&currentTime);
  conv->PostButtonEvent(MT_BTN_LEFT,MTBUTTON_UP,&currentTime);
}
```

4.2.2. Client-side components

Interactive Scene-graph Rendering is hosted by the GPAC MPEG player (part of GPAC multimedia solution package). Its functionalities are mapped to two blocks, namely Rendering and User Interaction Handler.

Rendering: The stream received through the down-link is decoded, the multimedia scene-graph objects and their semantics are recovered and classified into visual and non-visual content. The visual content is displayed by using the basic GPAC libraries. The non-visual content (user interaction sensors and JavaScript) are subsequently forwarded to the User Interaction Handler. While all the rendered content and the user event follows the MPEG-4 syntax, the GPAC libraries had to be modified in our study so as to handle the semantic management of the content, according to the *Scene-graph Manager*.

User Interaction Handler: This component has three main functionalities. First, by using the MPEG-4 interaction mechanisms, it captures the user events. Secondly, it makes a decision about processing the event locally (at the client-side) or remotely (at the server-side). In the former case, it computes the corresponding updated scene-graph, allowing the Rendering block to display it and then notifies the *Interaction Manager* about that action. In the latter case, it simply forwards the event to the *Interaction Manager* by one of the two mechanisms explained in Section 3.3. This module also required the modification of the GPAC reference software, so as to support the *ServerCommand* specified by the MPEG-4 standard but, to our best knowledge, not implemented yet (at least not in an open-source, publicly available, software).

4.2.3. Network components

Down-link: The traffic from server to client corresponds to live multimedia data; consequently, this channel is managed by the RTSP/RTP over UDP (Real Time Streaming Protocol/Real Time Protocol over User Datagram Protocol [39]). In our study, the use of the UDP was an implementation choice rather than a technical requirement; should the applicative environment impose constraints on the use of this protocol, alternative solutions can be considered, as the popular TCP [40] or as the emerging MMT (MPEG Media Transport) and DASH (Dynamic Adaptive Streaming over HTTP) MPEG standards [41], [42].

Up-link: This channel is mainly used by the client in order to enable server-side user interactivity, according to the MPEG-4 mechanisms, by exploiting both *AJAX HttpRequests* and the *ServerCommand*. The former case is supported by the HTTP in conjunction with TCP [43]. To the best of our knowledge, no study on the practical usage of the BiFS *ServerCommand* is reported in literature [44]; hence, we considered both the TCP and UDP when dealing with the latter case[5]. Note that as for the downlink, the protocol choice can be made according to the particular configuration in which the application is expected to work, without restricting the architectural generality.

5. Benchmark

The experiments were successively conducted so as to assess the four main properties of the MPEG-4 mobile thin client remote display: the visual quality of the rendered content (Section 5.1), the downlink bandwidth consumption (Section 5.2), the user interactivity efficiency (Section 5.3), and the CPU activity at the thin client side (Section 5.4).

These experiments were carried out on the following setup:

- *server*: a desktop platform, with Intel Xeon CPU, 3.2 GHz, 4GB of RAM, 5400rpm 500GB of HDD;
- *client*: an HTC HD2 smartphone, with Snapdragon™ CPU, 1GHz, 448MB of RAM, 768 MB internal memory;
- *network*: an USB Wi-Fi 802.11g access point directly connected to the server; the mobile client located at distance varying between 2 meters and 5 meters from the access point, with a direct line of sight.

While a large number of studies reported in the literature [45]-[47] already evaluated the MPEG technologies performances when serving all types of video content applications, the present study is oriented towards two real-life, native X window applications, namely the gEdit [48] text editor and the Epiphany www browser [49]. The former illustrates applications generating simple graphics, icons and text (development, office, e-mail, chat, *etc.*). The latter is an incremental stage, at which (high quality) images and more complex graphics are also generated; hence, the content generated by Epiphany is representative not only for the www browsing but also for image editing, virtual map accessing or professional medical applications, for instance.

The MPEG-4 based architecture presented in Figures 6 and 7 was implemented into three cases: (1) BiFS, (2) BiFS-Extended and (3) LASeR-Extended; the last two cases correspond to the extensions of the basic BiFS/LASeR remote display, obtained by using advanced multimedia scene management techniques (see Section 4.2 and Appendix 2). Note that BiFS-Extended and LASeR-Extended required the basic GPAC player to be adapted accordingly. In the sequel, these three MPEG-4 based solutions were benchmarked against on-the-market mobile thin client technologies: basic VNC, VNC-HEXTILE, and the Linux implementation of RDP [50].

The complete framework was assessed by carrying out two experiments. The gEdit text editing experiment considers 5 users, each of which typing for 5 minutes the text corresponding to the beginning of Plato's Republica. In order to investigate the case of web browsing, Epiphany was run by 5 users, each of which performing the following actions: (1) load Google

[5] The alternative usage of UDP and TCP with the AJAX HttpRequests for handling the user interaction was already presented in the authors' previous study [51]; however the experiments in [51] did not consider the MPEG-4 BiFS ServerCommand.

page, (2) type “Wikipedia mobile”, hit enter and wait for the page to be load, (3) click the Wikipedia mobile link and wait for the Wikipedia mobile page to be loaded, (4) type “chocolate” in the search area, hit enter and wait for the searched result page to be displayed, (5) click the link “bitter” and wait for the new page to load, (6) click the “Bookmark” menu item, select the google.news link, and wait for the page to load, (7) click the home icon, and wait for the www.debian.org home page to load, (8) scroll down, (9) click the “File” menu item and select “Quit”.

5.1. Visual quality

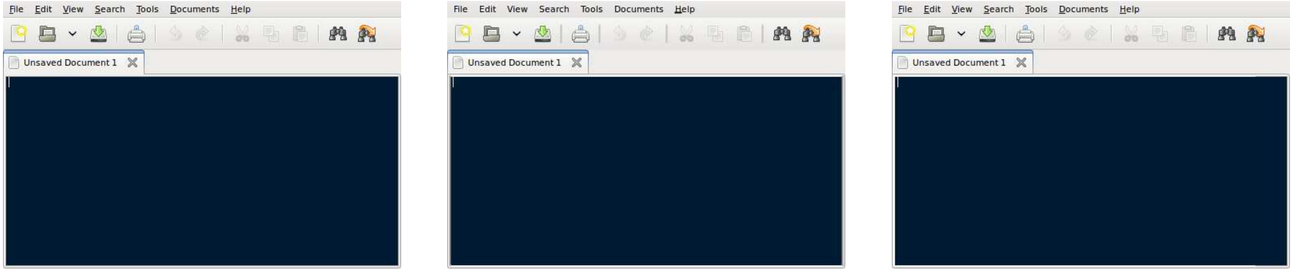
The content conversion from X11 to MPEG-4 BiFS/LASeR intrinsically introduces differences between the original and the converted visual representations. The aim of the present sub-section is to evaluate these conversion artifacts. In this respect, two approaches can be followed, i.e. the subjective and the objective evaluations.

5.1.1. Subjective evaluations

As the visual quality is a subjective concept, it cannot be assessed but by repeated tests, involving representative panels of human observers and testing conditions (device performances and visualization conditions, etc) as well as a large variety of content. To our best knowledge, no common ground for carrying out such an experiment in the case of text editing/www browsing graphical content is formalized today[6]. Consequently, in our case, the visual quality can be illustrated but not subjectively assessed. In this respect, we consider both screenshots represented in Figures 8 and 9, and video sequences corresponding to the content displayed on the thin client when performing the above-mentioned experiments; these video sequences can be downloaded from: *https://www.box.com/s/gch37qe9p6mtqzoli31e*.

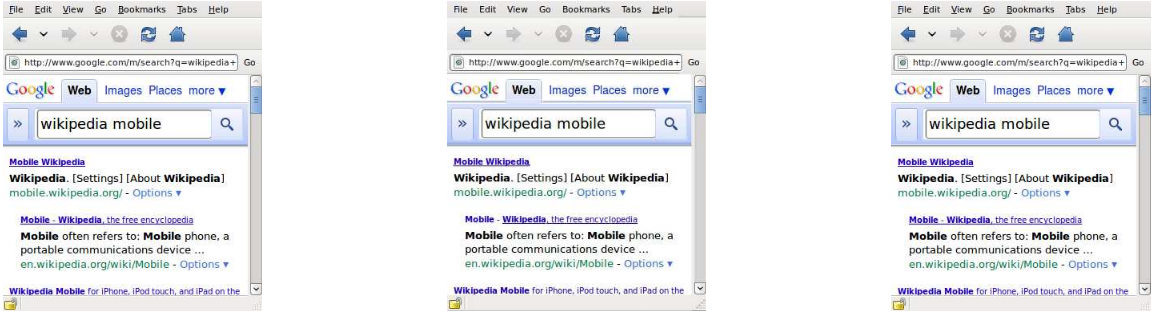
Figures 8 and 9 illustrate the quality of the MPEG-4 converted content, for the two above mentioned experiments. No illustration has been done for VNC, VNC-HEXTILE and RDP, as their server visual content is kept unchanged during the transmission and displaying; the visual content generated by the BiFS-Extended is identical to the one generated by the basic BiFS. Figures 8 and 9 show the type of differences induced by the MPEG conversion mechanism. For instance, in the text editing case, the lines separating the icons are different and the first letter in menu items are underlined with different width lines. The same line positioning/width errors may occur in the www browsing conversion.

[6] For other image processing applications, like the quality of television pictures, ITU elaborated recommendations (ITU-R BT 500-12 and BT 1438) [52] précising all the testing conditions and interpretation of the results.



(a) (b) (c)

Figure 8. Illustration of the text editing application run on the server (a) and displayed on the mobile thin client, after its conversion into BiFS / BiFS-Extended (b) and LASeR-Extended (c).


File Edit View Go Bookmarks Tabs Help
http://www.google.com/m/search?q=wikipedia+
Go
Google
Web
Images Places more ▾
»
wikipedia mobile

Mobile Wikipedia
Wikipedia. [Settings] [About **Wikipedia**]
mobile.wikipedia.org/ - Options ▾

Mobile - Wikipedia, the free encyclopedia
Mobile often refers to: **Mobile** phone, a portable communications device ...
en.wikipedia.org/wiki/Mobile - Options ▾

Wikipedia Mobile for iPhone, iPod touch, and iPad on the

(a) (b) (c)

Figure 9. Illustration of the www browsing application run on the server (a) and displayed on the mobile thin client, after its conversion into BiFS / BiFS-Extended (b) and LASeR-Extended (c).

5.1.2. Objective evaluations

The objective measures generally evaluate the differences between two images based either on the average differences among the pixels or on the correlation. In our experiments, Table I, we considered seven such measures: (1) *pixel difference based measures* (PSNR - peak signal to noise ratio, AAD - absolute average difference, and IF - image fidelity) and (2) *correlation based measures* (CQ - correlation quality, SC - structural content, NCC - normalized cross-correlations, and SSIM – structural similarity).

The identity between two images is expressed by the ideal values for these measures (PSNR → ∞, AAD = 0, IF =1, CQ = SC = NCC = SSIM = 1). Note that although no objective quality measure can guarantee the quality perceived by the human observer, they are commonly in use in image processing, [53], [54], [55], [56], [57], [58].

For the two experiments, in order to assess the visual quality, the rendered visual content corresponding to each and every scene update is converted into pixel maps and is subsequently saved in the ppm format on both server and client sides (thus obtaining pairs of images on which the objective measures are computed). In the case of the text editing experiment, one scene update is generated for each character typed by a user. Consequently, the number of images generated by each user in 5 minutes depends on his/her typing speed; in our experiments, we recorded 652, 827, 753, 694 and 798 characters for the five users, respectively. The related values presented in Table I (the gEdit columns) are obtained by averaging the visual quality measures obtained for each scene-update and for each user (*i.e.* are computed as average values on 3724 image pairs). As in the case of the www browsing experiments, one scene update is generated for each browsing step, each user generates 9 pairs of images; consequently, the related values presented in Table I (the Epiphany columns) are computed by processing 45 image pairs.

In order to offer statistically relevant information about the visual quality assessment, 95% confidence intervals were computed [59], [60]. For each experiment, each technology and each objective metrics, Table I presents the average value and the associated 95% error; hence, the corresponding 95% confidence intervals are given by $(average - error\ ;\ average + error)$.

In Table I, the PSNR average values (in dB) are approximated to the closest integer, the AAD, IF, CQ, SC and NCC average values are presented with 0.001 precision while a 0.000001 precision was chosen for the average value of SSIM. One more decimal digit was added in each case for the error presentation. Table I shows that, with singular exceptions (the PSNR and the SSIM values in the case of the Epiphany), the average values become statistical relevant even without considering their confidence limits: the 95% estimation error is lower than the precision to which the average values were filled-in in Table I[7].

The values corresponding to BiFS-Extended are identical to the basic BiFS ones. As the VNC, VNC-HEXTILE and RDP do not alter the visual quality, they result in ideal values for the considered measures.

In the objective visual quality assessment, we considered measures designed for natural images and not for heterogeneous visual content, combining text, graphics, icons, and images. This particularity in the content can justify some apparently contradictory values in Table I; for instance, in the case of the LASeR conversion of the gEdit, the best PSNR was obtained (42dB) but the related CQ is very low (0.702). When the content produced by the application is closer to natural images (*e.g.* the www browsing case) these discrepancies fade: for the LASeR conversion, PSNR = 40 dB and CQ = 0.953.

Table I. Visual quality evaluation for X11 to MPEG (BiFS, BiFS-Extended and LASeR-Extended) conversion.

	text editor (gEdit)				**www browser (Epiphany)**			
	BiFS / BiFS-Extended		**LASeR-Extended**		**BiFS / BiFS-Extended**		**LASeR-Extended**	
	average	*error*	*average*	*error*	*average*	*error*	*average*	*error*
PSNR (dB)	30	0.0	42	0.0	32	1.2	40	1.4
AAD	0.003	0.0000	0	0.0000	0.002	0.0008	0.005	0.0004
IF	0.998	0.0000	0.999	0.0000	0.999	0.0009	0.999	0.0001
CQ	0.929	0.0000	0.702	0.0000	0.974	0.0006	0.953	0.0003
SC	0.995	0.0000	1	0.0000	0.997	0.0005	1.009	0.0007
NCC	1	0.0000	0.999	0.0000	1	0.0004	0.995	0.0041
SSIM	0.999980	0.0000000	0.999999	0.0000000	0.999956	0.0000132	0.999992	0.0000031

[7] When computing the confidence intervals, the correlation between the images corresponding to successive scene updates was neglected; however, because of the very small variance of the values corresponding to each and every quality metric, the practical relevance of the results is not affected by this approximation.

5.2. Downlink bandwidth consumption

After the scene initialization, information is sent through the network downlink for each and every scene-update, be it initiated by the user (*e.g.* typing a letter or clicking) or by the server (*e.g.* a screen refresh). In the former case, the amount of traffic on downlink depends on the particular action they take (*e.g.* typing a letter will generate less traffic than clicking a menu item). In the latter case, the amount of traffic on downlink is random, depending on the server status and X application behavior.

For the text editing experiment, the values (in KBytes) of the bandwidth required by the corresponding cumulative downlink traffic, averaged over the 5 users, are plotted as a function of time (indexed in minutes) in Figure 10 (the value "0" on the abscissa refers to the scene initialization). Note that in this experiment, the number of scene updates varies with the scene updates generated by each user, *i.e.* with the number of letters they actually typed in each time interval (*e.g.*, after 5 minutes, 652, 827, 753, 694 and 798, respectively).

The www browsing experiment is illustrated in Figure 11, where the values (in KBytes) of the cumulative network traffic, averaged over the 5 users, are plotted (as a function of the 9 steps) in Figure 11. Note that this time the amount of traffic generated by each user is quite the same (each user generating the same updates) and small differences occurred only because of the server initiated downlink traffic.

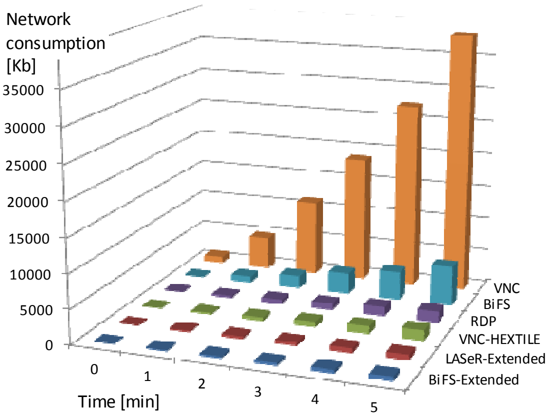


Figure 10. Average bandwidth consumption (in KBytes) for text editing (gEdit), as a function of time.

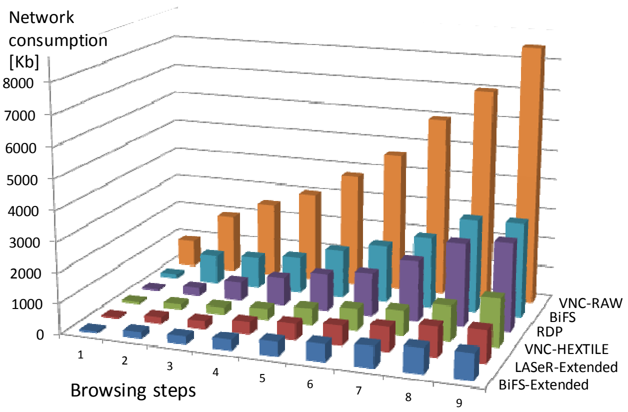


Figure 11. Average bandwidth consumption (in KBytes) for www browsing (Epiphany), as a function of the browsing step.

Figures 10 and 11 establish that for the two considered applications, BiFS-Extended is the best solution. In the text editing scenario, it outperforms LASeR-Extended, VNC-HEXTILE, RDP, basic BiFS and VNC by factors of 1.2, 2.3, 2.5, 9.3 and 60, respectively. When considering the www browsing, the BiFS-Extended gain over its competitors ranges from 1.2 to 10.

These compression gains are mainly due to two key factors the BiFS-Extended solution features. First, the visual content sent from the server to the client is no longer considered as a sequence of raw images (*i.e.* pixels) but as a collection of multimedia contents, semantically structured according to their types. This way, each type of content can be compressed with its optimal encoding mechanism. Secondly, the developed scene-graph management mechanism (illustrated in Appendix 2) eliminates the need for the retransmission of the visual content that was already sent to the client. Although the application periodically regenerates the same visual content (*e.g.* icons, user actions like "mouse over", *etc*.), the network will no longer be overcharged accordingly. By comparing the results concerning the BiFS-Extended to those corresponding to the BiFS, information about the practical impact of exploiting the semantic information in the scene management is obtained.

5.3. User interaction

As previously mentioned, the MPEG-4 BiFS standard makes provisions for two different ways of transmitting the user interactivity through the up-link: AJAX HttpRequest and *ServerCommand*. Consequently, in this section, the BiFS and BiFS-Extended will be considered in two different cases, according to their ways of exploiting the up-link.

In our study, we considered the two most frequent user events: keyboard strokes and mouse (pointing device) clicks.

The size of traffic generated through the up-link channel, as measured for each solution, is represented in Table II. These values depend on the technology but are independent with respect to the particular event (typing E or A generates the same traffic, right click generates the same traffic as the left click, *etc*.) and to the network conditions.

Table II also provides information about the network round-trip times, *i.e.* the time elapsed between the moment when the user interactivity actually takes place and the moment when the updated scene graph is displayed. As similar interaction mechanisms are obtained for keyboard strokes and mouse clicks, the related round-trip times are to be equal. However, these values slightly depend on the network conditions. The values presented in Table II are obtained as average values over all the users and all the 3894 events they generated: 3724 characters for gEdit, 125 characters for Epiphany (5 users typing "Wikipedia mobile" and "chocolate") and 45 clicks. The corresponding 95% confidence intervals featured errors lower than 1ms.

Table II. The size of the traffic generated through the back channel by elementary user events.

	TRAFFIC (bytes)		**ROUNDTRIP-TIME** (ms)
	KEYBOARD STROKE	**MOUSE CLICK**	**KEYBOARD STROKE / MOUSE CLICK**
VNC / VNC-HEXTILE	586	586	80
RDP	186	618	130
BiFS / BiFS-Extended / LASeR-Extended [AJAX HTTPRequest]	564	581	20
BiFS / BiFS-Extended [ServerCommand – TCP]	72	82	18
BiFS / BiFS-Extended [ServerCommand – UDP]	46	56	18

Table II shows that BiFS / BiFS-Extended solution considering the *ServerCommand* using UDP requires the lowest bandwidth, reaching 46 bytes (*i.e.* an up-link bandwidth gain factors from 4 to 12) for a keyboard stroke and 56 bytes (*i.e.* an up-link bandwidth gain factors from 10 to 11) for a mouse click, while keeping the interactivity round-trip times at 18ms. The same minimal round-trip times (18ms) are obtained for BiFS / BiFS-Extended considering the *ServerCommand* using TCP; however, with respect with the VNC/VNC-HEXTILE and RDP, the gains in the up-link bandwidth range now between 2.5 and 8. No clear advantage of the ServerCommand over the AJAX HTTPRequest has been identified by this experiment. Note: Table II reports only the values corresponding to the server-side interactivity, the most disturbing solution from the QoE point of view. Although supported by the architecture in Figures 6 and 7, the client-side interactivity evaluation is outside of the scope of this paper.

5.4. CPU activity

The amount of processor power needed to run the remote display client in order to render all the streamed content is assessed in this section. As from this point of view the relevant information is brought by the maximal CPU usage, in this experiment we considered only the www browsing application.

The measurements presented in Figure 12 are devoted to the values of the maximum CPU activity when browsing the www, according to the steps described in Section 5.2. It can be noticed that the remote display solutions that use raw pixel representation of the images (BiFS and VNC) produce less CPU activity than the rest (BiFS-Extended, LASeR-Extended and VNC-HEXTILE). However, it can be seen that the BiFS-Extended solutions does not exceed the maximal CPU activity of 58% (browsing step 7), compared with the LASeR reaching 68% (browsing step 9) and VNC-HEXTILE 95% (browsing step 7) of the total available computational resources on the device. This makes the BiFS-Extended solution even more appropriate for thin clients.

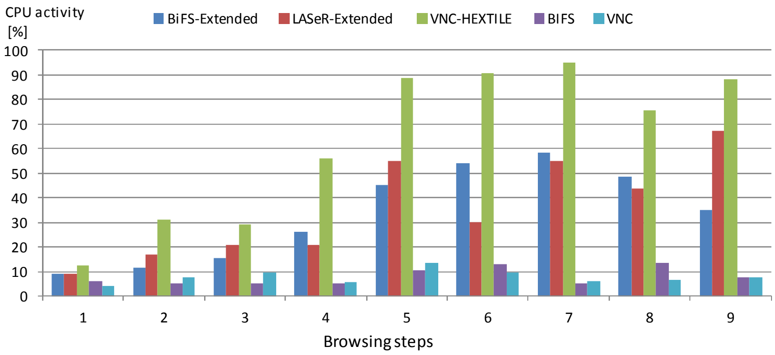


Figure 12. The average maximum CPU consumption (in %) while browsing, as a function of the browsing step.

Note that the RDP case is not represented in the Figure 12, as it is a solution integrated into the thin client Windows mobile OS and its accurate measurement is practically impossible to obtain. However, the experiments we carried out pointed to the fact that the RDP is the lightest solution.

6. Towards industrialization

While the previous section brings to light that the BiFS-Extended with *ServerCommand* interactivity support represents the best candidate (amongst the investigated solutions) in terms of the benchmarked constraints (visual quality, downlink consumption, real-time interactivity efficiency and computational activity), this section focuses on a way of reaching its industrial acceptance.

The novel solution is based on an open architecture, Figures 6 and 7. The way in which the modules are developed and located alleviates the need for the modification of the legacy software (be it OS or application) and allows a straightforward integration into emerging commercial application platforms, with minimal modification on both server and client sides. On the one hand, the server should be updated with the architectural framework while the rest of the applications can be kept unchanged. On the other hand, at the client, only an MPEG-4 player needs to be installed.

Such an approach completely satisfies the requirements of the mobile device switching. Firstly, all administration tasks are to be performed on the server-side: applications (*e.g.* www browsing) can be installed/updated/removed/replaced on the server without changing anything on the client. Secondly, the terminal independence is ensured by the MPEG-4 ISO standards. For instance, the GPAC framework is already available for use on most thin client terminals (Windows Mobile, Android and Apple iOS) and desktop computers (Windows, Linux and Mac OS). Moreover, it is able to play all types of MPEG multimedia content: audio, 2D, video, 3D BiFS, LASeR, VRML, SVG, 2GP and so forth. Note that the BiFS-Extended solution is perfectly compliant with the MPEG-4 BiFS standard; however, its development required the adaptation of the GPAC Framework, particularly concerning the *ServerCommand*. Thirdly, all components are supported by strong and rising open source communities, thus ensuring their potential evolution.

These three properties appeal to the various industrial players, from telco operators and service providers to third party software editors. The interest towards the architecture advanced in this paper is even broader in perspective, with the advent of cloud computing [61], and of modern distributed collaborative environments [62]-[65].

7. Conclusion and Future work

To the best of our knowledge, the paper advances the first semantic multimedia scene-graph remote display for mobile thin clients. In this respect, new architectural components are specified, designed and implemented, in order to provide an end-to-end, completely functional solution. The distinctive factors of this solution are: (1) visual quality expressed by seven objective metrics, *e.g.* PSNR values between 30 and 42dB or SSIM values larger than 0.9999; (2) downlink bandwidth gain factors ranging from 2 to 60; (3) real-time user event management ensured by network roundtrip-time reduced by factors of 4 to

6 and by up-link bandwidth gain factors from 3 to 10; (4) feasible CPU activity, larger than the RDP but reduced by a factor of 1.5 with respect to the VNC-HEXTILE.

This successful proof of concept of a semantic multimedia remote display for mobile thin clients opens the way to its integration into ready-to-use applications, fulfilling the expectations within real-life scenarios: terminal independence attained by ISO compliance (both on the client and server side), backward compatibility and open source support. Such overall properties make it a potential solution for use in cloud virtualization or distributed collaborative mobile environments. Of course, this study also requires the user quality of experience to be subjectively assessed: consequently, extending the ITU-R principles in so as to specify a subjective test procedure for this type of applications is also part of our future work.

As a final remark, the architecture presented in this paper is not restricted for use with the MPEG-4 scene description technologies. Consequently, research perspectives are connected to interactive multimedia scene-graph optimization under bi-directional error-prone network constraints and to the conceptual and functional synergies to be established among related yet different *de facto* and *de jure* standards like MPEG-4, Flash and HTML5 [66].

Appendix 1: Application panorama

In practice, the VNC and RDP technological supports are exploited by a large variety of ready to use applications. While an exhaustive list of such applications is practically impossible to be done and is also out of the scope of our paper, consider for instance:

- VNC-based applications: Apple Remote Desktop [67], Cendio ThinLinc [68], Chicken [69], ChunkVNC [70], Crossloop [71], EchoVNC [72], Ericom [73], Goverlan Remote Control [74], NoMachine NX [75], iTALC [76], KRDC [77], Mac HelpMate [78], N-central [79], noVNC [80], RealVNC [81], RapidSupport [82], Remote Desktop Manager [83], TigerVNC [84], TightVNC [85], TurboVNC[86], UltraVNC [87], X11vnc [88];
- RDP-based applications: AnywhereTS [89], Citrix XenApp [90], CoRD [91], DualDesk [92], Ericom [73], FreeRDP [93], NoMachine NX [75], KRDC [77], N-central [79], rdesktop [94], Remote Desktop Manager [83], Techinline [95], xrdp [96], XP/VS Server [97].

Moreover, proprietary (undisclosed) remote viewers' technological support and applications are also available. For instance, TeamViewer [98] exploits NAT (Network Address Translation) for establishing a connection, based on the RFB (VNC) and RDP protocols. GoToMyPC [99] exploits the Citrix ICA (Independent Computing Architecture) proprietary protocol, but also supports VNC (RFB) and RDP. PhoneMyPC [100] exploits its proprietary protocol, without exposing any specification detail. Oracle and Sun Microsystems offer Virtual Desktop Infrastructure (VDI) [101] solution based on the proprietary ALP – protocol (Appliance Link Protocol). Unfortunately, all these solutions are not open for research (undisclosed specification) and development (no source code); consequently, they cannot be objectively benchmarked in our research study.

All these applications may provide additional levels of functionalities, like built-in encryption, file transfer, audio support, multiple sessions, seamless window, NAT pass-through, IpV6 support, video or 3D. The study, reported in the present paper, is placed at the technological support level; consequently, the actual application peculiarities will not be further discussed.

Appendix 2: Semantic content management

When trying to optimize content transmission from server towards mobile thin clients, three directions can be exploited: *visual content re-usage*, *client memory control* and *network adaptability*.

Regardless its type (text editing, www browsing, …) each X Application can periodically generate identical *visual content*. Such a case does not only occur when refreshing the screen but also when dealing with some fixed items (frequent letters/words typing, icons or menus displaying, *etc*.) or with repeated user actions (mouse over, open, save, *etc*.). Consequently, significant bandwidth reduction is *a priori* likely to be obtained by reusing that content directly on the client side, instead of resending it through the network. However, in order to take practical advantage of this concept, a tool for automatically detecting the repetition of the visual content and for its particular management should be provided.

The thin *client memory* is limited by its hardware resources. While content re-use would suppose, as a limit case, the caching of all the visual content previously generated in a session, the limited memory resources requires a mechanism for dynamically adjusting the cached information. Several implementation choices are available, from a fixed time window to more elaborated decisions based on actual frequency of usage or on the content semantic.

For mobile clients, the network connection is prone to dynamic changes (bandwidth variation, random errors, *etc*.). Moreover, the mobile user might also want to switch the terminal during a work session (*e.g.* from a smartphone to a tablet). Consequently, the encoding parameters should be real-time adapted to the client/network conditions. Unfortunately, the nowadays MPEG-4 scene representation technologies do not provide a direct solution to this issue. Moreover, the flexibility requirement set on the *Compression & Transmission* block forced us to map this functionality to the *Scene-graph Manager* level.

In the sequel, a functional solution jointly addressing the three above-mentioned aspects is illustrated for the particular case of image content by the flowchart in Figure 13.

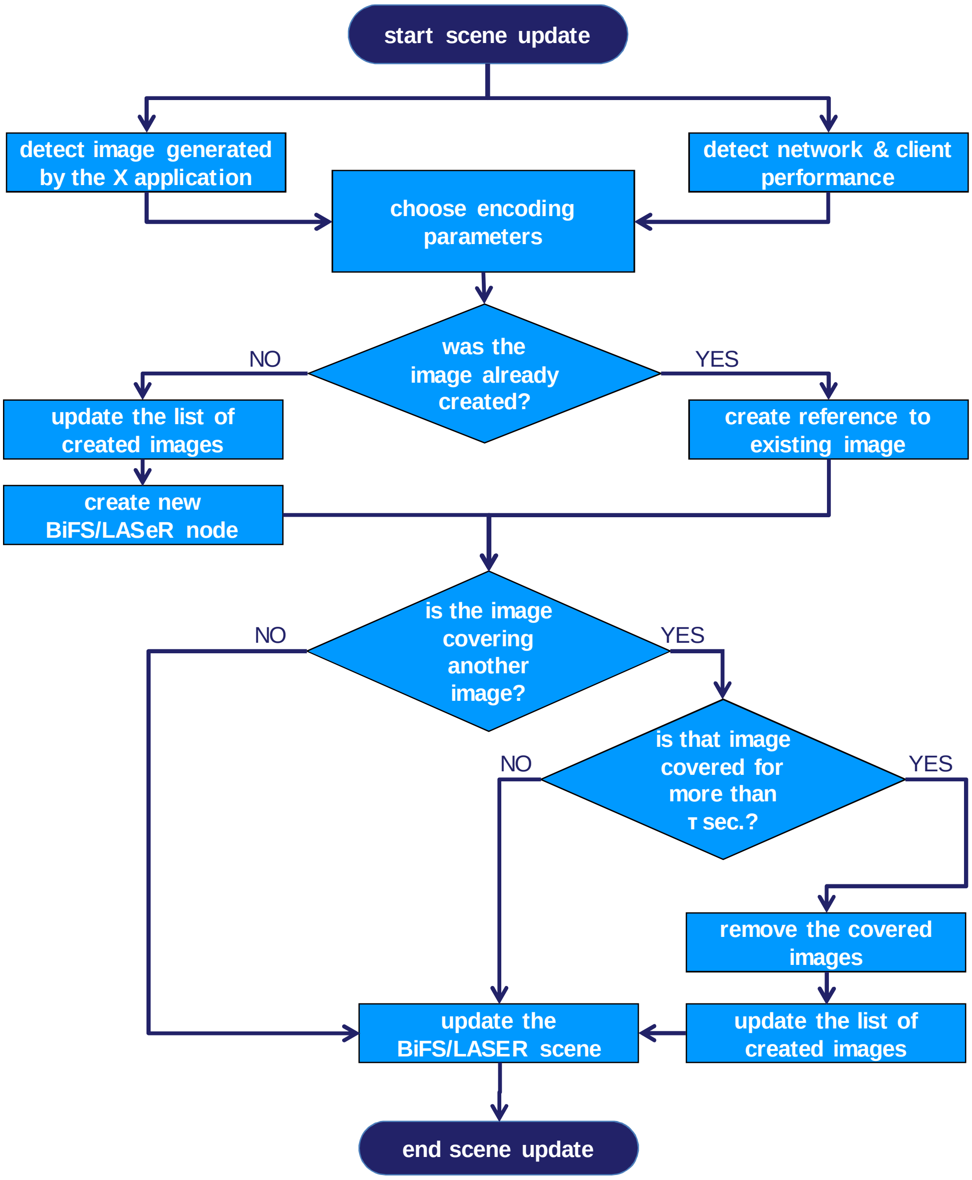

start scene update
detect image generated by the X application
detect network & client performance
choose encoding parameters
was the image already created?
NO
YES
update the list of created images
create reference to existing image
create new BiFS/LASeR node
is the image covering another image?
NO
YES
is that image covered for more than τ sec.?
NO
YES
remove the covered images
update the BiFS/LASER scene
update the list of created images
end scene update

Figure 13. Flowchart for image management.

The scene updating starts by detecting an image generated by the application and by reading some external network/client parameters. The image, its semantic information and the network/client parameters are combined so as to establish some encoding parameters (*e.g.* a low bandwidth and a low-resolution display would lead to a low quality factor for a JPEG compression).

Then, the existence of this image in the scene graph is checked. This task is achieved by computing the MD5 hash of the image and by searching it into a list containing the hashes of all images already used in the scene-graph. According to the way in which this list is organized (from a simple hash record to more sophisticated relations between hash, its usage, its time stamp, *etc.*) different functionalities can be provided by this module.

In the case when the image already exists in the scene, a simple reference (pointer) to the corresponding image is created. Otherwise, the hash record list is updated and the new image with its encoding parameters are placed in a new node (or, in several nodes) in the scene-graph.

As a side effect of this mechanism, the memory resources required by the client are increased. Hence, for thin clients, the image reusing should be restricted in time. In our implementation, we combined some temporal and spatial information about the cached images: assuming some images in the scene are not visible (*i.e.* they are covered by other visual elements) for more than τ seconds (in the experiments, $\tau = 180$ seconds), they are removed from the scene and the hash record list is updated accordingly. Of course, different decision making rules can be implemented here: while directly impacting the system performances, they would not affect the architecture generality.

Finally, the BiFS/LASeR scene is updated so as to take into account these changes: adding a new image / pointer to an image and remove some old images.

List of abbreviations

AAD	Absolute Average Difference
AJAX HttpRequest	Asynchronous JavaScript And XML HyperText Transfer Protocol Request
ALP	Appliance Link Protocol
AVC	Advanced Video Coding
BiFS	Binary Format for Scene
CPU	Central Processing Unit
CQ	Correlation Quality
ECMA	European Computer Manufacturer Association
FLV	FLash Video
GDI	Graphical Device Interface
HTML	HyperText Markup Language
IF	Image Fidelity
I/O	Input / Output
iOS	iPhone Operating System
ISO	International Organization for Standardization
LASeR	Lightweight Application Scene Representation
Mac OS	Apple Operating System

MPEG	Moving Picture Expert Group
NCC	Normalized Cross-Correlation
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
png	Portable Network Graphics
ppm	Portable Pixel Map
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
RDP	Remote Desktop Protocol
RFB	Remote FrameBuffer
RIM	Research In Motion
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SC	Structural Content
SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
SWF	ShockWave Flash
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VDI	Virtual Desktop Interface
VM	Virtual Machine
VNC	Virtual Network Computing
VRML	Virtual Reality Modeling Language
Wi-Fi	Wireless Fidelity
xHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XMT	eXtensible MPEG-4 Textual

Acknowledgement

This research was initiated by the MobiThin project and has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement no. 216946 and is partly founded by the CloudPort French national project.

References

[1] D. Schlosser, A. Binzenhofer, B. Staehle, “Performance Comparison of Windows-based Thin-Client Architectures”, 2007 Australasian Telecommunication Networks and Applications Conference, Christchurch, New Zealand, December 2 – 5 2007

[2] P. Simoens, P. Praet, B. Vankeirsbilck, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, P. Demeester, “Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices”, *ATNAC 2008*, December 7-10 2008

[3] X Window System description, http://www.x.org

[4] VNC, Virtual Network Computing description, http://www.realvnc.com

[5] NX, NoMachine description, http://www.nomachine.com/

[6] RDP, Microsoft - Remote Desktop Protocol: Basic connectivity and graphics remote specification, http://msdn.microsoft.com/en-us/library/cc240445

[7] K-J. Tan, J-W. Gong, B-T. Wu, D-C. Chang, H-Y. Li1, Y-M. Hsiao, Y-C. Chen, S-W. Lo, Y-S. Chu, J-I. Guo, “A Remote thin client system for real time multimedia streaming over VNC”, ICME 2010, Singapore, July 19-23, 2010

[8] H. Shen, Y. Lu, F. Wu, S. Li, “A High-Performance Remote Computing Platform”, PerCom 2009, Galveston, Texas, March 9-13, 2009

[9] P-F. Yasser, A-H. Kambiz, H. Alnuweiri, “Internet Delivery of MPEG-4 Object-Based Multimedia” IEEE MultiMedia, April–June 2004

[10] C. Lethanhman, H. Isokawa, T. Kato, “Multipath Data Transmission for Wireless Thin Clients” *UBICOMM 2009*, Sliema, Malta, October 11-16 2009

[11] M. Bannan, D. Kellogg, “All about android”, Nielsen, 2011

[12] Yan Lu, Shipeng Li, Huifeng Shen, “Virtualized Screen: A Third Element for Cloud Mobile Convergence”, IEEE MultiMedia, Volume 18, Issue 2, April 2011.

[13] MPEG-4 BiFS standard specification, ISO/IEC JTC1/SC29/WG11 14496-11

[14] Asadi M.K., Dufourd J.-C, “Context-Aware Semantic Adaptation of Multimedia Presentations”, IEEE International Conference on Multimedia and Expo, 6-8 july 2005, Amsterdam, Holland.

[15] E. Izquierdo, J.R. Casas, R. Leonardi, P. Migliorati, Noel E. O’Connor, I. Kompatsiaris, M. G. Strintzis, “Advanced content-based semantic scene analysis and information retrieval: the chema project”, Workshop on Image Analysis for Multimedia Interactive Services, 9-11 April 2003, London, UK.

[16] Aaron Carroll , Gernot Heiser, “An Analysis of Power Consumption in a Smartphone”, USENIX 2010, June 22–25, 2010

[17] Microsoft® RemoteFX™, http://technet.microsoft.com/en-us/library/ff817578(WS.10).aspx

[18] “Forecasts for smartphone operating system sales until 2015”, Gartner, www.rossdawsonblog.com

[19] S. Battista, F. Casalino, C. Lande, “MPEG-4: A Multimedia Standard for the Third Millennium, Part 1”, IEEE MultiMedia, pp: 74 - 83, January–March 1999

[20] S. Battista, F. Casalino, C. Lande, “MPEG-4: A Multimedia Standard for the Third Millennium, Part 2”, IEEE MultiMedia, pp:76-84, January–March 2000

[21] MPEG-4 LASeR standard specification, ISO/IEC°JTC1/SC29/WG11 14496-20

[22] J-C. Dufourd, O. Avaro, C. Concolato, ”An MPEG Standard for Rich Media Services”, IEEE MultiMedia, pp: 60-68, October–December 2005

[23] Adobe Flash, http://www.adobe.com/

[24] The Java Language Specification, http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html

[25] W3C, Scalable Vector Graphics Specification, http://www.w3.org/TR/2002/WD-SVG12-20021115

[26] W3C, Timed Text Markup Language Specification, http://www.w3.org/TR/ttaf1-dfxp/

[27] xHTML specification, http://www.w3.org/TR/xhtml-basic/

[28] GZIP file format specification version 4.3, rfc 1952, http://tools.ietf.org/html/rfc1952

[29] Z. Liu, Y. Saifullah, M. Greis, S. Sreemanthula, “HTTP Compression Technique”, Wireless Communications and Networking Conference, 2005 IEEE, pp. 2495 - 2500 Vol. 4, New Orleans, USA, March 13-17 2005

[30] J. Song, B.-D. Lee, “Mobile Rich Media Technologies: Current Status and Future Directions”, KSII Transactions on Internet and Information Systems, Vol. 5, No. 2, February 2011

[31] Ecma-262, EcmaScript documentation, http://www.ecmascript.org/docs.php

[32] E.J. Bruno, “Ajax: Asynchronous JavaScript and XML”, *Dr. Dobb's Journal*, v 31, n 2, February 2006, p 32-35

[33] M. Mitrea, P. Simoens, B. Joveski, I. J. Marshall, A. Tanguengayte, F. Preteux, B. Dhoed, “BiFS based approaches to remote display for mobile thin clients”, in Proceedings of SPIE, Vol: 7444, 74440F (2009); doi:10.1117/12.828152, San Diego, August 2009

[34] B. Joveski, M. Mitrea, F. Preteux, “MPEG-4 LASeR - based thin client remote viewer”, EUVIP2010 - European Workshop on Visual Information Processing, Paris, July 2010

[35] B. Joveski, P.Simoens, L.Gardenghi, J.Marshall, M. Mitrea, B. Vankeirsbilck, F. Prêteux, B. Dhoed, “Towards a multimedia remote viewer for mobile thin clients”, in Proceedings of SPIE, Vol: 7881, 788102 (2011); doi:10.1117/12.876279, 2011, San Francisco, January 2011

[36] GPAC – Open Source Multimedia Framework, http://gpac.sourceforge.net/index.php

[37] C. Concolato, J. Le Feuvre, J.-C. Moissinac, “Design of an Efficient Scalable Vector Graphics Player for Constrained Devices”, IEEE Transactions on Consumer Electronics, Vol. 54, No. 2, May 2008

[38] Live555 Streaming Media, source code libraries, www.live555.com

[39] UDP – User Datagram Protocol, RFC: 768, http://tools.ietf.org/html/rfc768

[40] TCP – Transmission Control Protocol specification, RFC: 793, http://www.ietf.org/rfc/rfc793.txt

[41] MPEG Media Transport (MMT), http://mpeg.chiariglione.org/working_documents/mpeg-h/mmt/mmt_co.zip

[42] MPEG Dynamic Adaptive Streaming over HTTP (DASH), http://mpeg.chiariglione.org/working_documents/mpeg-dash/MPEG-DASH-Tutorial.pdf

[43] Hypertext Transfer Protocol specification, RFC2616, http://www.http-compression.com/rfc2616.txt

[44] GPAC publication page, http://gpac.wp.institut-telecom.fr/about/references/

[45] Beg, M.S., A. Muslim, Y. C. Chang , T. F. Tang , “Performance evaluation of error resilient tools for MPEG-4 video transmission over a mobile channel”, Conference on Personal Wireless Communications, 15-17 Dec. 2002

[46] Galluccio, L. , “Transmission of adaptive MPEG video over time-varying wireless channels: modeling and performance evaluation”, IEEE Transactions on Wireless Communications, Volume: 4 , Issue: 6 Page(s): 2777- 2788, November, 2005

[47] Basso, A. Kim, Y.-J., Jiang, Z. ,“Performance evaluation of MPEG-4 video over realistic EDGE wireless networks”, The 5th International Symposium on Wireless Personal Multimedia Communications, Volume: 3 Page(s): 1118-1122 vol.3, 2002

[48] gEdit official text editor for the GNOME desktop environment, http://projects.gnome.org/gedit/

[49] Epiphany web browser for the GNOME desktop environment, http://projects.gnome.org/epiphany/

[50] XRDP – Open source project for Remote Desktop Protocol for Linux version 0.4.2, http://sourceforge.net/projects/xrdp/files/xrdp/0.4.2/

[51] B. Joveski, L. Gardenghi, M. Mitrea, F. Prêteux , “Towards collaborative MPEG-4 BiFS mobile thin remote viewer”, 15th IEEE International Symposium on Consumer Electronics, ISCE2011, Singapore, 14-17 June 2011

[52] Recommendation ITU-R BT.500 , “Methodology for the subjective assessment of the quality of television pictures”, 09/2009, ITU-T Tutorial, Video Quality Experts Group (VQEG), “Objective perceptual assessment of video quality: Full reference television” 2004

[53] ISO ISO/IEC JTC1/SC29/WG11 MPEG2010/N11275 Dresden DE, Vittorio Baroncini, Jens-Rainer Ohm, Gary J. Sullivan, “Report of Subjective Test Results of Responses to the Joint Call for Proposals (CfP) on Video Coding Technology for High Efficiency Video Coding (HEVC)”, April 2010

[54] Adel Rahmoune, Pierre Vandergheynst, Pascal Frossard, “Flexible motion-adaptive video coding with redundant expansions”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 2, February 2006

[55] Athanassios Skodras, Charilaos Christopoulos, Touradj Ebrahimi, “The JPEG 2000 still Image compression standard”, IEEE Signal Processing Magazine, September 2001

[56] Giovanni Petrazzuoli, Marco Cagnazzo, Beatrice Pesquet-Popescu, “High order motion interpolation for side information improvement in DVC”, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP, April 2010

[57] Hsien-Po Shiang, Mihaela van der Schaar, “Multi-User Video Streaming Over Multi-Hop Wireless Networks: A Distributed, Cross-Layer Approach Based on Priority Queuing”, IEEE Journal on Selected Areas in Communications, vol. 25, no. 4, May 2007

[58] Francesca De Simone, Matteo Naccari, Marco Tagliasacchi, Frederic Dufaux, Stefano Tubaro, Touradj Ebrahimi, “Subjective Quality Assessment of H.264/AVC Video Streaming with Packet Losses”, EURASIP Journal on Image and Video Processing, Volume 2011

[59] T.C. Fry, “Probability and Its Engineering Use”, D van Nostrand, 1965

[60] R.E. Walpole and R.H. Myers, “Probability and Statistics for Engineers and Scientists”, 4th ed., MacMillan Publishing, NY, 1989

[61] J. Baliga, R.W.A. Ayre, K. Hinton, R.S. Tucker, “Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport”, Proceedings of the IEEE, Vol. 99, No.1, January 2011

[62] I.J. Marshall, M. Mitrea, B. Joveski, L. Gardenghi F. Preteux, “Requirements on MPEG Scene Technology for Collaborative Applications” , ISO/IEC JTC 1/SC 29/WG 11, No: N12206, Torino, Italy, July 18-23 2011

[63] I.J. Marshall, M. Mitrea, B. Joveski, L. Gardenghi F. Preteux, “Context and Objectives for MPEG Scene Technology with Collaborative Applications”, ISO/IEC JTC 1/SC 29/WG 11, No: N12207, Torino, Italy, July 18-23 2011

[64] I.J. Marshall, M. Mitrea, B. Joveski, F. Preteux, “Use Cases for Collaborative Applications”, ISO/IEC JTC 1/SC 29/WG 11, No: N12208, Torino, Italy, July 18-23 2011

[65] I.J. Marshall, M. Mitrea, B. Joveski, F. Preteux, “Draft Call for Proposals for Scene Technologies for Collaborative Applications”, ISO/IEC JTC 1/SC 29/WG 11, No: N12209, Torino, Italy, July 18-23 2011

[66] HTML5 working draft by W3C, http://www.w3.org/TR/html5/

[67] Apple Remote Desktop, http://www.apple.com/remotedesktop/

[68] Cendio ThinLinc, http://www.cendio.com/products/thinlinc/

[69] Chicken, http://sourceforge.net/projects/cotvnc/

[70] ChunkVNC, http://www.chunkvnc.com/

[71] Crossloop, http://www.crossloop.com/

[72] EchoVNC, http://sourceforge.net/projects/echovnc/

[73] Ericom, http://www.ericom.com/

[74] Govern Remote Control, http://www.pjtec.com/

[75] NoMachine, http://www.nomachine.com/

[76] iTALC, http://italc.sourceforge.net/

[77] KRDC, http://www.kde.org/applications/internet/krdc/

[78] MAC HelpMate, http://www.machelpmate.com/

[79] N-central, http://www.n-able.com/products/n-central/

[80] noVNC, http://kanaka.github.com/noVNC/

[81] RealVNC, http://www.realvnc.com/

[82] RappidSupport, http://www.rapidsupport.net/pages/screenshots.php

[83] Remote Desktop Manager, http://remotedesktopmanager.com/

[84] TigerVNC, http://sourceforge.net/apps/mediawiki/tigervnc/index.php?title=Main_Page

[85] TightVNC, http://tightvnc.com/

[86] TurboVNC, http://www.virtualgl.org/Downloads/TurboVNC

[87] UltraVNC, http://www.uvnc.com/

[88] X11vnc, http://www.karlrunge.com/x11vnc/

[89] AnywhereTC, http://anywherets.com/

[90] Citrix XenApp, http://www.citrix.com/English/ps2/products/product.asp?contentID=186

[91] CoRD, http://cord.sourceforge.net/

[92] DualDesk, http://www.dualdesk.com/features/index.html

[93] FreeRDP, http://www.freerdp.com/

[94] RDesktop, http://www.rdesktop.org/

[95] Techinline, http://techinline.com/

[96] Xrdp, http://www.xrdp.org/

[97] XP/VS Server, http://www.thinstuff.com/faq/index.php?action=artikel&cat=9&id=97&artlang=en

[98] TeamViewer, http://www.teamviewer.com/en/index.aspx

[99] GoToMyPC, http://www.gotomypc.com/remote_access/remote_access

[100] PhoneMyPC, http://softwareforme.com/?page_id=6

[101] Virtual Desktop Interface by Oracle,http://www.oracle.com/us/technologies/virtualization/oraclevm/061153.html