Towards a multimedia remote viewer for mobile thin clients

B. Joveski[*1] P.Simoens[2] L.Gardenghi[1] J.Marshall[3]
M. Mitrea[1] B. Vankeirsbilck[2] F. Prêteux[4] B. Dhoed[2]

[1]Institut TELECOM / TELECOM SudParis / ARTEMIS Department, France
[2]IBBT & Ghent University, Belgium
[3]Prologue Software, France
[4]MINES ParisTech
[5]UPMC Université Pierre et Marie CURIE - Sciences et Médecine

ABSTRACT

Be there a traditional mobile user wanting to connect to a remote multimedia server. In order to allow them to enjoy the same user experience remotely (play, interact, edit, store and share capabilities) as in a traditional fixed LAN environment, several dead-locks are to be dealt with: (1) a heavy and heterogeneous content should be sent through a bandwidth constrained network; (2) the displayed content should be of good quality; (3) user interaction should be processed in real-time and (4) the complexity of the practical solution should not exceed the features of the mobile client in terms of CPU, memory and battery. The present paper takes this challenge and presents a fully operational MPEG-4 BiFS solution.
Key words: remote viewer, multimedia, mobile thin clients, BiFS.

1. INTRODUCTION

A thin client can be considered as a device featuring only displaying capabilities (and very few -if any- computing capabilities). Such a device can work only within a client-server configuration and intrinsically exploits the underlying remote viewer. In its largest acceptation, a remote viewer regroups all the software mechanisms making it possible for the content computed on the server to be displayed on the client and for the user interaction captured on the client to be processed by the server.

In practice, traditional remote viewers are represented as a three component solution: *the server* processing the heterogeneous multimedia content (text, graphics, image, video, 3D, …), *the network* transferring the content, and *client* displaying the content, Figure 1. On one hand, the server is in charge of computation of all the heterogeneous graphical content generated by its active applications; it collects their graphical output, converts it into binary compressed format and transmits it to the client. On the other hand the client is in charge of collecting all the user inputs (keystrokes, mouse click/move, …) and transmit them back to the server.

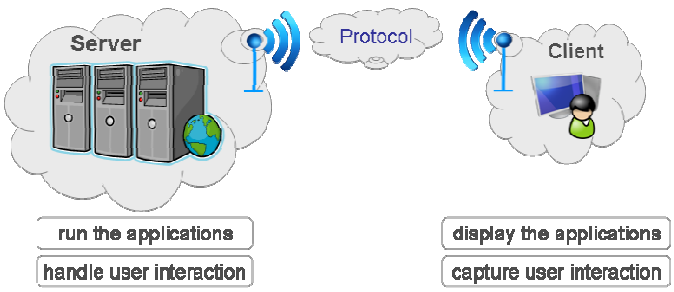

Figure 1 A remote display software solution.

Nowadays, in order to develop remote display solutions for fixed wired environments, several sound technologies are available: X [1], NX [2], pTHINc [3], VNC [4] to mention but a few. However, significant differences arose when considering the ways in which a remote viewer may be implemented in mobile environment: heavy and

[*] Corresponding author: Bojan Joveski, Telecom SudParis, 9, rue Charles Fourier, 91011 Evry, France
bojan.joveski@it-sudparis.eu, www.it-sudparis.eu/artemis, www.mobithin.eu

heterogeneous content should be sent through a bandwidth constrained network, the displayed content should be with a good quality, user interaction should be processed in real-time and the complexity of the practical solution should not exceed the features of the mobile client in terms of CPU, memory and battery. Moreover, each potential solution is practically restricted by its genericity, *i.e.* the possibility of its deployment irrespective of the thin terminal particularities.
Within the existing solutions it is practically impossible to jointly meet all the requirements for mobile (wireless) thin client scenarios. In order to bridge this gap, the present paper follows a different approach: it demonstrates that the MPEG-4 BiFS scene technology can alone provide for all the needs of a remote mobile thin viewer.

2. STATE OF THE ART

One of the important challenges for creating a remote viewer architecture is to decide at which level to capture the graphics generated by the applications running on the server. Depending on the level at which the graphical content is captured, different kinds of information are available for further processing (*e.g.* encoding of the content). To encode the display updates, existing remote display solutions typically use low-level graphic instructions or video codecs. The first approach is adopted by conventional remote display architectures. For instance, Citrix XenApp [5] and MicroSoft Remote Desktop Services [6] apply elementary drawing commands and images, whereas Virtual Network Computing (VNC) updates the display by filling rectangular screen areas with bitmaps. These low-level graphic instructions are the most efficient way to remotely display applications that only update a small portion of the display and have a slow refresh rate, such as office applications. In turn, video codecs are more convenient when the graphics exhibit a high level of detail and large parts of the display are frequently updated. This approach is mainly used for multimedia applications and remote 3D rendering, such as virtual 3D environments or 3D medical imaging applications [7, 8, 9].
The encoding format is however only optimized for a specific subset of applications. Using the inappropriate format to compress application graphics leads to high bandwidth requirements and degraded visual experience, *e.g.* because static displays are encoded as video frames or because text characters are encoded by lossy image compression techniques [10]. To support a wider range of applications with a single remote display framework, hybrid approaches have been presented that integrate multiple encoding formats. For example, Citrix Speedscreen Acceleration forwards video streams in their original format to the client, while the other parts of the display are still encoded through the Citrix proprietary protocol. This approach is only possible for video streams for which the appropriate video codec is installed at the thin client. Similarly, Tan et al. [11] divide the display in low- and high-motion regions that are encoded respectively by means of VNC drawing primitives and MPEG-4 AVC (a.k.a. H.264) [12] frames.
These hybrid techniques provide no adequate solution to compress the complex scenes of contemporary applications, comprising objects with highly diverse graphic characteristics, such as text, images, widgets and embedded audio and video. A better approach is to encode each object individually, with its own optimal encoding format. However, this requires that the client is provided with the appropriate information on the different objects, such as their position and encoding format, in addition to the binary encoded objects themselves. In this respect, the MPEG-4 BiFS description language standard [13] has already proved its potential [14].

3. MPEG-4 BiFS *vs.* IMAGES

Moving Picture Experts Group (MPEG) introduced the MPEG 4 standard (ISO/IEC 14496 - Coding of audio-visual objects) as a collection of representation and compression methods, for dealing with video, audio, graphics and text coding formats. Under this framework, MPEG 4 defines a dedicated description language, called Binary Format for Scene (BiFS) which is able to describe the heterogeneous content of the scene, to manage the scene object behavior (e.g. object spin) and to ensure the timing of conditional updates (e.g. user input/interactivity).
The novelty of BiFS not only refers to the scene description but also to the scene compression. Traditionally, the heterogeneous visual content was represented by successive frames composing a single video to be eventually compressed by some known codecs (such as MPEG-2 [15] or MPEG 4 AVC). BiFS follows a completely different approach: it allows each object to be encoded with its own optimal coding scheme (video is coded as video, text as text, and graphics as graphics).
The MPEG-4 BiFS standard itself offers several mechanisms for handling images on the scene. They are usually considered a texture |for a given scene object and are described by: *ImageTexture*, *PixelTexture* and *CacheTexture*. Depending on how the images would be used for creating one complex heterogeneous scene, the usage of the three image texture is as follows:

- *ImageTexture*: This node is used when the image is introduced to the scene by using URL (link) to an existing image (being it located on the server or the client device). Note that the existence of an image file is required.

- *PixelTexture*: This node has the power of describing the image on the scene as a string of R G B pixels. It means that it can represent all the images in a non compressed manner; the image file existence is not required.
- *CacheTexture*: This node is meant to use compressed images (png, jpeg) on the scene. For each and every image a cached file might be created on the device for further re-usage of the image content if and as needed. Depending on the usage of this node, the image file existence is not required but it may be considered.

In our previous work [14], we already proved the potential benefits from using BiFS technologies in order to implement a multimedia remote viewer. The present study reconsiders the initial architecture and improves on it in order to: (1) reduce the downlink bandwidth consumption by advanced image management, (2) make provision for heterogeneous scene transmission and (3) ensure real-time user-event processing.

4. ARCHITECTURAL FRAMEWORK

An original mobile thin client remote display architecture accommodating the MPEG-4 BiFS technology and devoted to the X Window System was synoptically presented in [14], and progressively extended with two new components: BiFS tree manager and User Event handling. This extended framework provides an optimization towards the thin client requirements and completes the remote viewer solution, Figure 2.

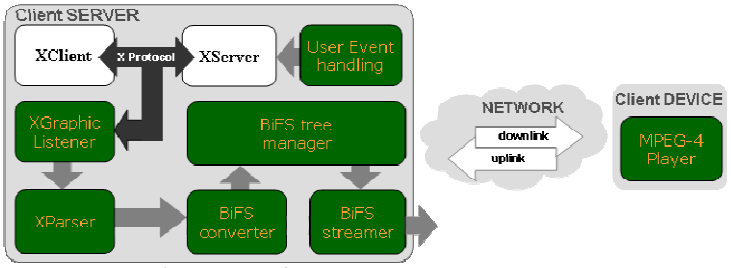

Figure 2. MPEG-4 based architectural framework for a mobile thin client remote display.

Server components:
The *XClient*, the *XServer* & the *XProtocol*: The *XServer*, *XClient* and the *XProtocol* are part of the X Window System, where the *XClient* is a generator of all the graphical primitives needed for further processing. They correspond to a traditional application, i.e. they are kept unchanged in the present study. The *XGraphic Listener* listens to the content generated by the *XClient* and further passed to the *XParser* which parses the XProtocol, and analyses the structure of the content. The BiFS converter maps a function to each X request that converts the X graphical content into MPEG-4 BiFS and then The BiFS streamer is used for real-time streaming of the converted contend to the mobile thin client using RTP/RTSP.
The User Event handling is a new component for receiving and handling of the user interaction (key pressed, mouse moved/clicked, …) sent from the mobile client device towards the server. After receiving the interaction the server is updated.
The BiFS tree manager as a new component into the framework is located between the BiFS converter and the BiFS streamer. This component has the ability to manage all of the BiFS converted heterogeneous content and of creating one complex MPEG-4 BiFS scene. It is not only responsible for building the hierarchy of the scene tree, but is also in charge of controlling how the content will be made available to the client, enhancing the user experience by reducing the network traffic and keeping the visual quality. In order to reduce the traffic, lossy and lossless compression of image content has been considered. The complete workflow for how images are managed by this component is described in Figure 3.

Client component:
The MPEG-4 interactive player renders the heterogeneous MPEG-4 content and captures the user.

Network components:
The network ensures the traffic from the server to the thin client (downlink), and vice-versa (uplink). In our framework RTP/RTSP was considered.

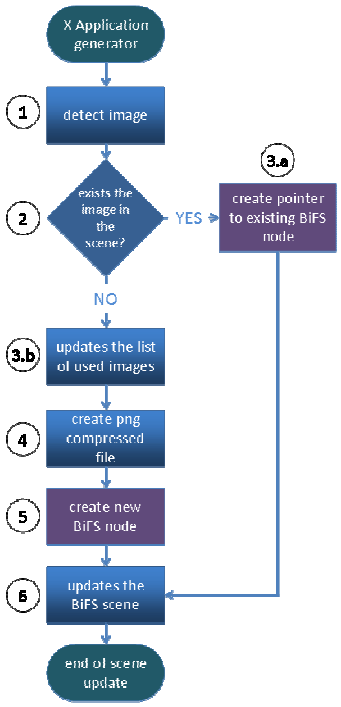

Figure 3. Flow chart of the BiFS tree manager.

The X Application generator as a part of the X Window System, represents the application running on the server generating the heterogeneous content needed for further processing (about 95% are images). After the content has been converted, the BiFS tree manager (1) detects the images generated by the X Applications, (2) compares every image with the list of previously generated images, according to a HASH mechanism, (3.a) for an existing image, it creates a pointer to an existing BiFS node and continues from point (6), (3.b) for a new image, it updates the list of images, (4) creates a compressed image file (png, jpeg, …), (5) creates new BiFS scene node, (6) updates the BiFS scene with the new node.

5. EXPERIMENTAL VALIDATION

The experiments were successively conducted so as to assess the four main properties of the MPEG-4 remote display: the visual quality of the displayed content (Section 5.1), the bandwidth consumption (Section 5.2), the user inter-action efficiency (Section 5.3), and the computational complexity required of the thin client (Section 5.4).

5.1. Visual quality

We focus on objectively assessing the visual quality of elaborated use cases, concerning different types of elementary content (such as www browsing and text editing), illustrated in Figure 4, 5 and Table 1,2.
Two types of measures have been considered, Table 1: (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 c*ontent, and NCC - *normalized cross correlation*).
In order to see which image compression can cope best with each of our use cases, for this experiment we considered three image compression mechanisms: (1) png, (2) jpeg with 75% of quality and (3) jpeg with 90% of quality.
In the www browsing scenario we considered the following actions: (1) open Epiphany application with google mobile as web page, (2) type “wikipedia mobile” and hit enter and wait for the new page to be loaded, (3) click on the “wikipedia” link and load the new web page, (4) type chocolate and hit enter and wait for the new page to load, (5) click on “taste” link and wait for the page to load, (6) open the “www.debian.com” link and wait the page to load, (7) open the link “http://farm1.static.flickr.com/66/227776718_90cf12451c.jpg” and wait the page to load. The experimental results are illustrated in Figure 4. Table 1 represents the averaged values by considering the content of each step compared with its original. Note that as in Step 7 the original and the converted images were identical in the png case, the corresponding PSNR would be infinite and it was not take into account when computing the average.
In the text editing scenario using the gEdit application, we considered typing the first paragraph from the Plato’s Republica. The visual quality results are illustrated in Figure 5 and detailed in Table 2.

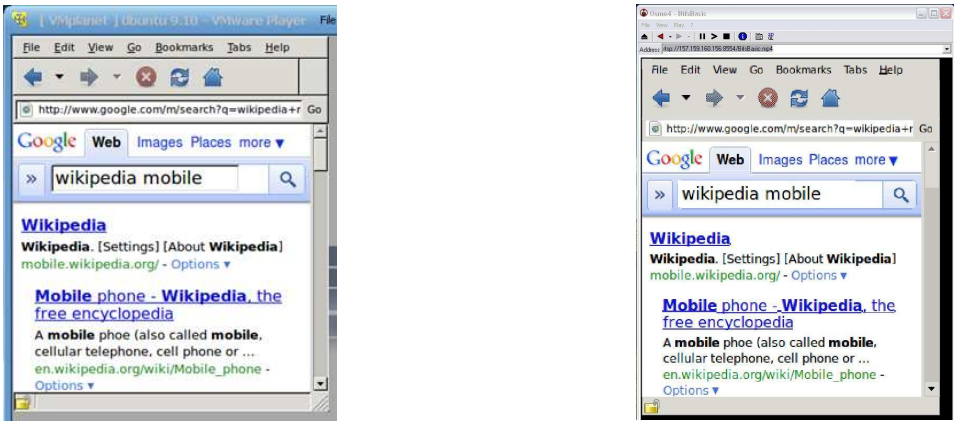

Figure 4. The browser application (Epiphany) run on the server (left) converted into BiFS content and displayed on the thin client (right).

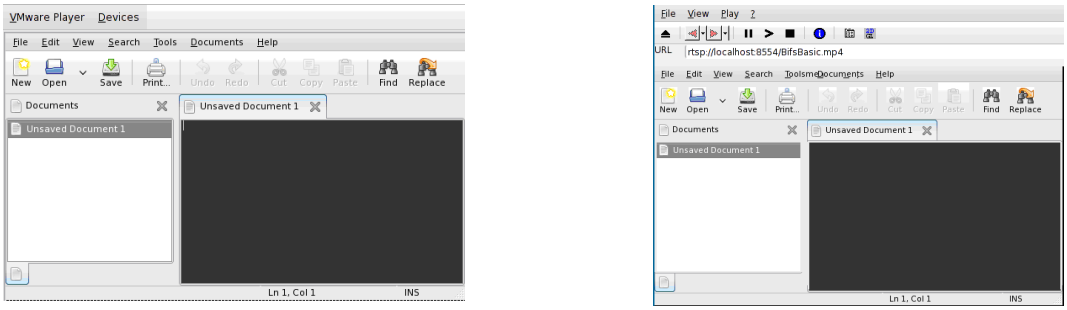

Figure 5. The text editor application (gEdit) run on the server (left) converted into BiFS content and displayed on the thin client (right).

Table 1. Visual quality evaluation for X11 to MPEG-4 BiFS conversion by averaging the converted content when using Epiphany.

	BiFS-PNG	BiFS-JPEG90	BiFS-JPEG75
PSNR	35.17	25.92	24.77
SC	1002	1019	1022
AAD	0.002	0.017	0.021
IF	0.998	0.994	0.993
Ncc	0.997	0.988	0.986
CQ	0.914	0.905	0.903

Table 2. Visual quality evaluation for X11 to MPEG-4 BiFS conversion by averaging the converted content when using gEdit.

	BiFS-PNG	BiFS-JPEG90	BiFS-JPEG75
PSNR	∞	17.86	17.54
SC	1000	1037	1051
AAD	0.000	0.088	0.100
IF	1.000	0.981	0.979
NCC	1.000	0.972	0.965
CQ	0.924	0.898	0.891

When analyzing the results in Tables 1 and 2, several conclusions can be drawn. As expected, the best visual quality is obtained when using the PNG compression. When the content is preponderantly composed by images (www browsing), the objective measures lead to quite good values for even jpeg compression. This is not the case when considering applications with heavy graphics (text editing). However, for such a content, low values for the quality metrics (e.g. PSNR = 17.86dB) do not necessary mean low quality for the visual content (see the example in Figure 5).

5.2. Bandwidth consumption

5.2.1. Downlink

The experiments were focused on evaluating the bandwidth required for www browsing or text editing.

In this respect, 6 types of solutions for the representation of content for remote display are considered: the basic VNC (RAW), the optimized VNC (HEXTILE), the standard BiFS using raw images, the standard BiFS using jpeg image compression and the standard BiFS technology using png image compression. All the MPEG-4 experiments are provided in two different settings: first, a basic version which does not consider the image re-using (hence, no alternative way 3.b in the flowchart in Figure 3) and then with this mechanism is implemented by using a HASH mechanism.

In order to investigate the case of web browsing, the Epiphany browser was used by 5 users, each of which performing the following actions: (1) open the Epiphany browser and load the Wikipedia mobile page, (2) type “chocolate”, hit enter and wait for the page to be load, (3) click “bitter” link and wait for the page to load, (4) click browser’s back button and wait for the page to load, (5) click “Mexico” link and wait for the page to be load, (6) click browser’s back button and wait for the page to load. The averaged (over the 5 users) values of the total bandwidth consumption are plotted in Figure 6.

The text editing experiment was carried out considering the gEdit application and 5 users, each of which executing the following actions: (1) open the gEdit, (2) click on the menu “File->file1.txt” open the first file (file with saved first page of Plato’s Republica), (3) select the first paragraph, (4) click on the menu “Edit->Copy”, (5) click on the menu “File->file2.txt” (an empty file), (6) click on the menu “Edit->Paste”, (7) click on the menu “File->Save”, (8) click on the menu “File->Close”, (9) click on the menu “File->Close”. The averaged (over the 5 users) values of the total bandwidth consumption are plotted in Figure 7.

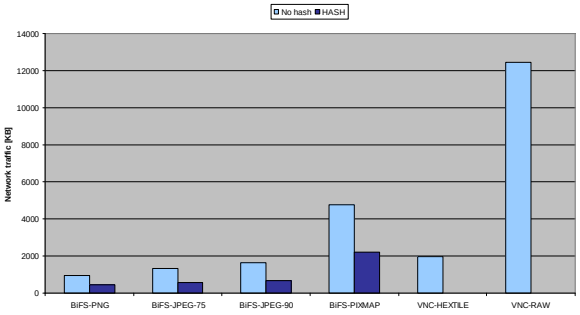

Figure 6. The network traffic required when using the epiphany (www browsing application)

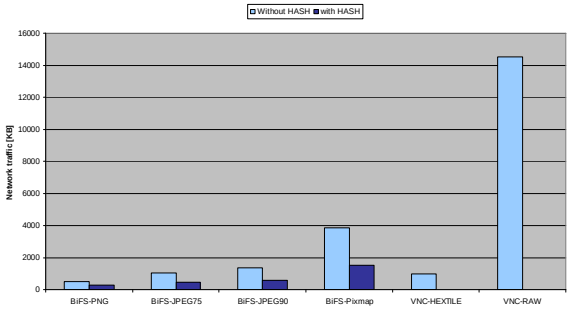

Figure 7. The network traffic required when using the gEdit (text editing application)

By analyzing Figure 6 and Figure 7, we can easily conclude that by using the HASH method for image re-usage, we reduce the network traffic by about 50% of the bandwidth required by each of the proposed solutions. Note that VNC is a solution where this HASH method can not be applied. A particular behaviour concerning image compression should also be noted: the images generated by these two types of applications are better compressed by the png mechanism than by jpeg mechanisms. This is a consequence of both their size and of their content.

5.2.2. Uplink

The illustrations in this section consider only the server side interaction.

Figure 8 exemplifies the interactivity loop for a key stroke: when the user strikes a key the corresponding ASCII code is generated at the scene level. This code is forwarded to the XServer, as an AJAX HTTP request. Consequently, the X Application is updated and the corresponding information is parsed, converted as an MPEG-4 scene update, streamed to the player and finally displayed. This network roundtrip time is less then 20ms (in the considered WiFi setup).

Figure 8. The client-server-client interactivity loop.

Although very efficient from the interactivity time point of view, the current MPEG-4 players are not yet optimized in terms of uplink bandwidth consumption: beyond the event information per se, additional protocol information is required for the opening and closure of the connection and for acknowledgements, etc. The example above (pressing the A key) generates 564 bytes of traffic, Table 3. Just to stress the point, the sentence "The quick brown fox jumps over the lazy dog" would require 24252 bytes. The same situation is encountered for a mouse click (note that mouse button pressing and releasing generates two different requests). Such a situation can be eliminated by using the ServerCommand though this is not yet supported by the most intensively used BiFS players.

Table 3. The size of the traffic generated through the back channel by elementary user events.

	Uplink (bytes)	**Downlink** (bytes)
Mouse click (down click only)	581	299
Keyboard press	564	299

Note that the values reported in Table 3 were measured for the BiFS based remote display. However, as BiFS and LASeR have similar interaction mechanisms, they can be considered as representative for both of the two technologies.

5.3. CPU consumption

The amount of processor power needed to run the Osmo4 player in order to render the converted graphical content is assessed. The Asus MyPal A639 (Intel Xscale processor 416 MHz, 1GB of ROM) is considered as the mobile thin client.

The measurements presented in Table 4 correspond to the initialization phase of two above mentioned applications: Epiphany and gEdit. Figure 9 is devoted to the time dependency of the maximum CPU consumption when browsing the www, according to the steps described in Section 5.2.

By individually running the two applications (Epiphany, gEdit) the maximum CPU consumption can be measured. However, in order to obtain values independent with respect to the other processes running on the same system, each experiment was run 5 times and the corresponding average values are presented in Table 4. These results show that even for simple office-like applications (gEdit), the HEXTILE VNC requires as much as 85% of the considered CPU, i.e. double the BiFS requirements.

A quite similar behavior can be noticed in Figure 9 which is devoted to a comparison between the maximal CPU consumption required in order to update the thin client content when browsing the www: the average BiFS gain is this time around 20%. These results can be explained by the extensive image rendering process required by VNC.

Table 4. The average maximal CPU consumption (in %).

	BiFS-OPTIMIZED	**VNC-HEXTILE**
Epiphany	35 %	77 %
gEdit	41 %	85 %

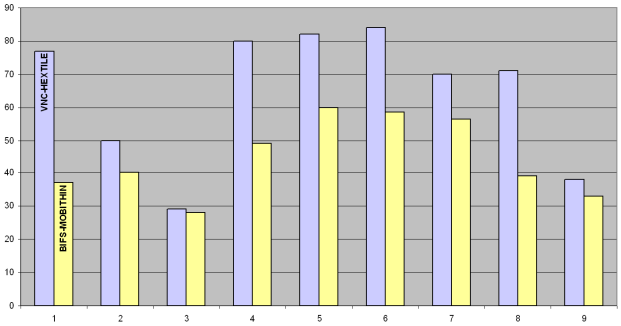

Figure 9. The average maximum CPU consumption (in %) while browsing, as a function of time indexed by the browsing steps).

6. CONCLUSION

By reconsidering the PoC presented in a previous study, the present paper presents the first fully functional multimedia oriented remote viewer. It is based on the conversion of the native X content into MPEG-4 BiFS for use by a BiFS tree manager allowing basic image transmission control. The remote viewer has been validated through www browsing (Epiphany) and text editing (gEdit) application. The quality of the visual content was objectively assessed by using both, pixel difference and correlation based measures. The evaluation of the downlink bandwidth consumption brought to light an overall gain of 98% with respect to the traditional VNC RAW and of about 70% with respect to optimized VNC HEXTILE.

Further work will be performed towards optimizing this remote viewer framework so as to be able tocope with the constrains of collaborative and distributed medical application data, cf. the ITEA2 European project MEDUSA.

7. ACKNOWLEDGEMENT

This research was done for the MobiThin project and has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement no. 216946.

8. REFERENCE

[1] X Window System, http://www.x.org.
[2] FreeNX: Freenx, http://freenx.berlios.de.
[3] Joeng Kim, Ricardo A. Baratto, Jason Nieh, “pTHINC: a thin-client architecture for mobile wireless web”, WWW2006, Edinburg, Scotland, May 23-26, 2006.
[4] T. Richardson, Q. Stafford-Fraser, K. R. Wood, A. Hopper, “Virtual Network Computing”, IEEE Internet Computing, 1998.
[5] Citrix Systems Inc.: http://www.citrix.com.
[6] Microsoft MicroSoft: Remote desktop protocol: Basic connectivity and graphics remoting specification. http://msdn.microsoft.com/en-us/library/cc240445.
[7] Paravati, G., Celozzi, C., Sanna, A., Lamberti, F. “A Feedback-Based Control Technique for Interactive Live Streaming Systems to Mobile Devices”, IEEE TRANSACTIONS ON CONSUMER ELECTRONICS 56(1), 190-197 (2010).
[8] Preda, M., Villegas, P., Moran, F., Lafruit, G., Berretty, R.P., ”A model for adapting 3D graphics based on scalable coding, real-time simplification and remote rendering”, VISUAL COMPUTER 24(10), 881-888 2008). International Conference on Cyberworlds, Hannover, GERMANY, OCT 24-27, 2007.
[9] Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P., Scopigno, R.:Protected interactive 3D graphics via remote rendering. ACM TRANSACTIONS ON GRAPHICS 23(3), 695-703 (2004).
[10] Simoens, P., Praet, P., Vankeirsbilck, B., De Wachter, J., Deboosere, L., De Turck, F., Dhoedt, B., Demeester, P., “Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices”, ATNAC: 2008 AUSTRALASIAN TELECOMMUNICATION NETWORKS AND APPLICATIONS CONFERENCE, pp. 391-396 (2008).

[11] Tan, K.J., Gong, J.W., Wu, B.T., Chang, D.C., Li, H.Y., Hsiao, Y.M., Chen, Y.C.,Lo, S.W., Chu, Y.S., Guo, J.I., “A remote thin client system for real time multimedia streaming over VNC” In: 2010 IEEE International Conference on Multimedia and Expo (ICME), pp. 992-7 (2010).
[12] MPEG-4 AVC: ISO/IEC JTC1/SC29/WG11 14496-10.
[13] MPEG-4 BiFS: ISO/IEC JTC1/SC29/WG11 14496-11.
[14] M. Mitrea, P. Simoens, B. Joveski, J. Marshall, A. Taguengayte, F. Prêteux, B. Dhoed, “BiFS-based approaches to remote display for mobile thin client”, SPIE, Vol. 7444, 74440F (2009); San Diego, CA, USA, August 2-5, 2009.
[15] MPEG-2 ISO/IEC 13818-2.