

## D5.4 – Phase I Experimental validation – Lab Trials

<b>Deliverable Type *</b>	: PU
<b>Nature of Deliverable **</b>	: O (Demo)
<b>Version***</b>	: <b>Released</b>
<b>Created</b>	: <b>30 June 2009</b>
<b>Contributing Workpackages</b>	: WP5
<b>Editor</b>	: <b>Bart Dhoedt (IBBT)</b>
<b>Contributors/Author(s)</b>	: IBBT(L. Deboosere, F. Azmat Ali, B. Vankeirsbilck, P. Simoens, B. Dhoedt), Prologue (A. Taguengayte), NTUK (B. Lecroart, F. Fok), IMEC (R. Torrea Duran), IT (B.Joveski, M. Mitrea, F. Prêteux) DTAG (F-J. Westphal, M. Kind)
<b>File Name</b>	: [MobiThin_D5.4_WP5_30June09_IBBT_V1.0]

- **\*Deliverable type:**
  - *PU = Public,*
  - *RE = Restricted to a group of the specified Consortium,*
  - *PP = Restricted to other program participants (including Commission Services),*
  - *CO = Confidential, only for members of the MobiThin Consortium (including the Commission Services)*
  
- **\*\* Nature of Deliverable:**
  - *P= Prototype,*
  - *R= Report,*
  - *S= Specification,*
  - *T= Tool,*
  - *O = Other.*
  
- **\*\*\*Version:**
  - *Preliminary,*
  - *Draft 1, Draft 2,...,*
  - *Released*

### Abstract:

**This deliverable describes lab trials carried out to validate the architecture and selected components of phase 1 of the MobiThin project. These trials include the optimization of the image transmission, based on MPEG BiFS/Laser content description, the evaluation of a thin client viewer deployed on a mobile phone, and demonstration of protocol adaptivity.**

**This deliverable comes in parallel with D5.3, describing the validation work carried out based on emulations.**

*“The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216946”*

**Keyword List: Mobile thin clients, experimental evaluation, adaptivity, MPEG image transmission, MPEG Laser**

The **MOBITHIN Project Consortium** groups the following Organizations:

Interdisciplinary Institute for BroadBand Technology vzw	IBBT vzw	BE
Deutsche Telekom AG	DTAG	DE
Prologue Software	Prologue	FR
Interuniversitair Micro-Electronica Centrum vzw	IMEC vzw	BE
NEC Technologies (UK) Ltd	NTUK	UK
Groupe des Ecoles des Télécommunications	GET	FR
JCP-Consult SAS	JCP	FR

## EDITORIAL Change Management

Version	Date	Editor/Author	Comments
Draft	June 3, 2009	Bart Dhoedt	Template created
D1	June 3, 2009	Bart Dhoedt	Pasted in section on protocol adaptivity (contributions from IBBT and NTUK)
D2	June 3, 2009	B. Lécroart	Added Mobile Device Integration of a Thin Client contribution from NTUK.
D3	June 11, 2009	Bart Dhoedt	Added sections 2(intro) and 3(scope)
D4	June 24, 2009	F.-J. Westphal	Internal review
Released	June 29, 2009	Bart Dhoedt	Final editing.

## Contents

<b>1.</b>	<b><i>Executive summary</i></b>	<b>4</b>
<b>2.</b>	<b><i>Introduction</i></b>	<b>5</b>
	<b>2.1 Project context</b>	<b>5</b>
	<b>2.2 Relation with D5.1 and D5.3</b>	<b>5</b>
<b>3.</b>	<b><i>Scope of the deliverable</i></b>	<b>6</b>
<b>4.</b>	<b><i>Experimental reports on Lab Trial</i></b>	<b>7</b>
	<b>4.1 “We have a problem”</b>	<b>7</b>
	4.1.1 Software/hardware architecture	7
	4.1.2 Lab trial set-up (hardware description – characteristics)	7
	4.1.3 Results and interpretation	8
	4.1.4 Conclusions – requirements met	10
	<b>4.2 MPEG based image transmission</b>	<b>10</b>
	4.2.1 Software/hardware architecture	10
	4.2.2 Lab trial set-up (hardware description – characteristics)	11
	4.2.3 Results and interpretation	13
	4.2.4 Conclusions – Requirements met	16
	<b>4.3 Mobile Device Integration of a Thin Client</b>	<b>17</b>
	4.3.1 Software/hardware architecture	17
	4.3.2 Ocean software architecture	18
	4.3.3 Lab trial set-up	19
	4.3.4 Description of experiments performed and the results	19
	4.3.5 Conclusions – Requirements met	20
	<b>4.4 Protocol adaptivity – Hybrid Streaming</b>	<b>21</b>
	4.4.1 Software architecture	21
	4.4.2 Lab trial set-up	22
	4.4.3 Description of experiments performed	24
	4.4.4 Results and interpretation	24
	4.4.5 Conclusions – Requirements met	27
	<b>4.5 Protocol Adaptivity - A Scheduling Approach</b>	<b>29</b>
	4.5.1 Software architecture	29
	4.5.2 A scheduling-based screen update transmission pattern	29
	4.5.3 Lab trial set-up	31
	4.5.4 Results and interpretation	32
	4.5.5 Conclusions – Requirements met	34
<b>5.</b>	<b><i>Overall conclusions</i></b>	<b>35</b>
<b>6.</b>	<b><i>References</i></b>	<b>40</b>
	<b>Appendices</b>	<b>41</b>
	<b>Appendix 1 Detailed comparison between existing thin client protocols</b>	<b>41</b>
	<b>A1.1 Machine Specifications</b>	<b>43</b>
	<b>A1.2 Reference Machine Computation Table</b>	<b>43</b>
	<b>A1.3 Bandwidth Measurement Data</b>	<b>43</b>
	<b>A1.4 CPU Load Measurement Data</b>	<b>44</b>
	A1.4.1 Raw Data	44
	A1.4.2 Corrected Data	44
	<b>A1.5 Memory Footprint Measurement Data</b>	<b>45</b>
	<b>Appendix 2 Power consumption of the Asus Eee PC</b>	<b>45</b>

# 1. EXECUTIVE SUMMARY

This report, together with its accompanying deliverable 5.3, brings together the validation results obtained in the first project phase of MobiThin. The results addressed are obtained through lab trials, i.e. validation work where the “real” hardware is used in building the different Proof-of-Concept demonstrations. The results presented related to the characterization of existing thin client protocols (analyzing their shortcomings), to the MPEG based image transmission and user interaction, to porting the thin client viewer to a mobile platform and to assessing two different adaptivity mechanisms.

The characterization of existing thin client protocols reveals important shortcomings, especially when multimedia applications are used (e.g. browsing combined with viewing multimedia content). Issues both relating to client CPU load (upto 41%), bandwidth (up to 80 Mbps for full screen MPEG4 viewing) as well as interactivity are reported.

An MPEG BiFS<sup>1</sup> based image transmission subsystem has been realized, compatible with a MPEG client viewer. Important savings in bandwidth usage (typically a factor of 5) have been observed, whilst keeping the quality level and offering interactivity. Efforts are underway to standardize this approach in the MPEG community.

The project also succeeded in porting a thin client viewer to a mobile phone, hence showing that thin client viewers are a workable option for remotely accessing application through a resolution limited device. Conclusions regarding usability were drawn (pointing devices, resolution, selecting user sessions and establishing a session with the servers).

Two different approaches were presented and evaluated in this report:

1. The hybrid streaming protocol, switching dynamically between H.264 streaming and a pixel-based format, depending on the content shown. The protocol automatically chooses the optimal transmission mode, based on the content characteristics. The adaptivity succeeds in realizing the optimal operation of the individual modes, resulting in lower client side CPU consumption and bandwidth usage.
2. The scheduled update approach allows the client and server to agree on an optimal display update frequency, adapting this frequency to the current network conditions. It was shown that the bandwidth used by the protocol can be very well controlled by judiciously tuning this update frequency.

The lab trials reported here have been constructed in accordance with the architectural documents and PoC design documents. The results obtained allow to conclude that the approach taken on each sub-domain addressed is justified, and hence realizing the PoC's is an important step in building the MobiThin E2E system.

---

<sup>1</sup> BiFS stands for Binary Format for Scenes, standardized as the part 11 of the MPEG-4 standard (ISO/IEC JTC1/SC29/WG11 14496-11); it allows heterogeneous multimedia content (graphics, video, audion, text, 3D, ...) to be represent into a unique scene.

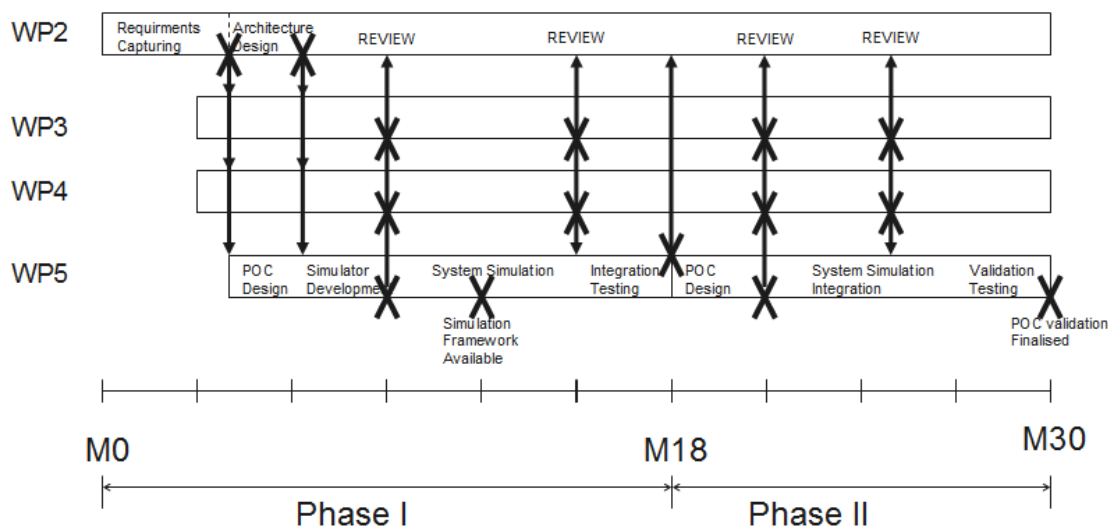
## 2. INTRODUCTION

### 2.1 PROJECT CONTEXT

*Common text for D5.3 and D5.4*

The MobiThin project is organized in two phases, allowing an incremental approach to realize its final goals. The deliverables D5.3 and D5.4 conclude the first phase, and the work reported is essential to effectively prioritize and plan the activities of the second project phase.

Work package 5 of the MobiThin project is to design, develop and evaluate proof-of-concept demonstrators in order to validate the approach taken in the project (e.g. on the architectural or algorithmic level). After the design (D5.1, M6) of these PoC's, tool development (D5.2), several PoC's were indeed realized, based on various components realized in WP3 and WP4 (Fig. 1). The findings of the validation work are reported in D5.3 and D5.4 and, in addition to show-casing and dissemination, results will mainly be used to guide and fine-tune the activities of the second project phase. While D5.3 focuses on emulation demonstrators (part of the hardware is emulated in software), D5.4 reports on lab-trials (using the intended hardware for the project).



**Fig. 1. High-level MobiThin project structure.**

### 2.2 RELATION WITH D5.1 AND D5.3

*Common text for D5.3 and D5.4*

The following demonstrators were defined in D5.1:

1. Wireless medium optimization (estimate the energy and consumption of the wireless components )
2. Image transmission based on MPEG (BiFS & LAsER) content representations (investigate how thin client systems can benefit from using an MPEG (LAsER, BiFS) standard multimedia player for graphic primitive transmission)
3. Demonstration: “We have a problem...” (demonstrate that without optimization, the currently available thin client protocol have important weaknesses)
4. Mobile Device Integration of a thin client (demonstrate a running thin client application on a mobile terminal with 3G or Wifi.)
5. Protocol Adaptivity (prove that adaptive protocols are necessary to cope with the changes in the environment and that this leads to better QoE and efficient bandwidth usage and/or CPU loads.)

6. User Management (show the ability of the Service Management Framework (SMF) o manage user configuration and user session management)
7. Session Migration (demonstrate the ability to automatically move a “persistent server side session from one thin client server to another with minor impact on the user’s QoE.)
8. Remote Devices/peripherals (demonstrate the use of peripherals connected on the thin client by the thin client server)

It the course of the project, it became apparent that the design and development of the Service Management Framework required more resources than anticipated, and therefore the PoC 8 (Remote devices/peripherals) was shifted to phase II of the project.

All other PoC’s have been successfully realized, and are reported on in D5.3 and 5.4. While 5.3 concentrates on emulation oriented PoC (where at least one component of the PoC is emulated –i.e. does not run on the actually intended hardware, but uses an emulation thereof instead-), D5.4 reports on lab-trials (using the “real” hardware for the set-up). The mapping of the PoC’s defined in D5.1 and the reporting is further clarified in Table 1.

PoC number	Title	Report (section)
1	Wireless medium optimization	D5.3 (4.1)
2	Image transmission based on MPEG	D5.4 (4.2)
3	“We have a problem...”	D5.4 (4.1)
4	Mobile Device Integration of a Thin Client	D5.4 (4.3)
5	Protocol Adaptivity	D5.4 (4.4 and 4.5)
6	User Management	D5.3 (4.3)
7	Session Migration	D5.3 (4.2)
8	Remote Devices/peripherals	Shifted to Phase II

**Table 1. Mapping of D5.1 PoC’s to reporting in D5.3/4.**

### 3. SCOPE OF THE DELIVERABLE

As outlined above, this deliverable focuses on reporting lab-trial results obtained on the PoC’s. These lab trials include:

1. “We have a problem” (demo carried out using WiFi link, using thin client hardware) (see section 4.1 – also presented as demo at the first NEM-summit and during the first review meeting)
2. MPEG-based image transmission (see section 4.2)
3. Mobile device integration of a thin client (Integration of a thin client protocol on a Mobile Phone “Ocean”) (demo also shown during the first review meeting) (see section 4.3)
4. Protocol adaptivity (adaptation demonstrated on actual thin client hardware) (two mechanisms are presented, see sections 4.4 and 4.5)

Each PoC report (detailed in section 4 of this document) has the same base structure: (i) a recap on the hardware/software architecture, (ii) presentation of the hardware set-up, (iii) presentation of obtained results and interpretation, (iv) conclusions (where appropriate, feedback to the requirements outlined in D2.1 is given).

## 4. EXPERIMENTAL REPORTS ON LAB TRIAL

### 4.1 “WE HAVE A PROBLEM”

#### 4.1.1 Software/hardware architecture

As illustrated in Figure 1, a classic thin client protocol transfers user data and screen updates between the MobiThin Client and the MobiThin Server. In this case an existing thin client and server are used.

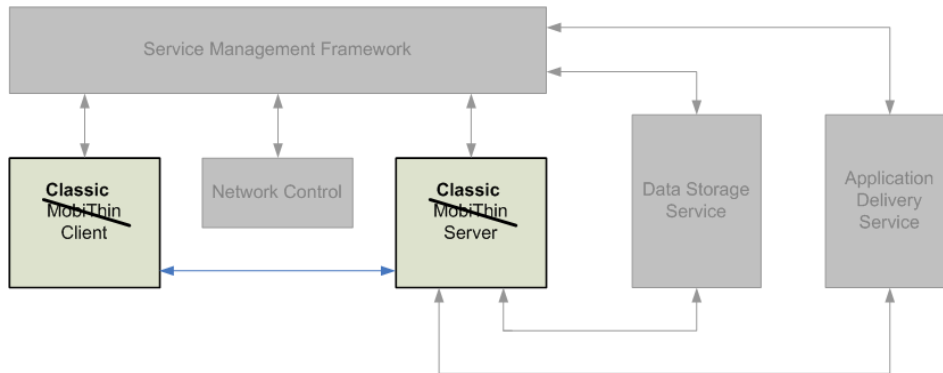


Figure 1 – We have a problem: components involved

In these lab trials we show that there is a problem with currently available technology, but the results will be used for other lab trials as reference.

#### 4.1.2 Lab trial set-up (hardware description – characteristics)

Figure 2 shows the components and devices involved for performing lab trials on “we have a problem”. As said, the protocol operates between the Thin Client and the Thin Client Server. On the Thin Client machine, a classic Thin Client viewer software must be installed. For testing purposes, components are to be present that generate events in order to have controlled tests. The same logic is applied to the Thin Client Server, where the Thin Client server counterpart software is installed, along with the necessary software components to perform the tests. Between the Thin Client Server and Thin Client a Network Impairment node is responsible for controlling e.g. the amount of packet loss or delay incurred by the network.

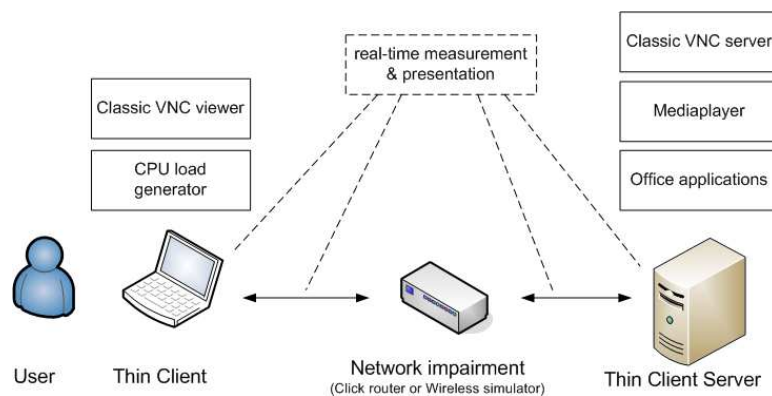


Figure 2 – We have a problem: lab trial hardware set-up

The details of the exact hardware the tests were performed on, are given together with the results in the next section, because not all tests were executed on the same equipment.

## 4.1.3 Results and interpretation

### 4.1.3.1 Visual aspects

A first problem that has been identified with five different existing thin client protocols, is that they do not handle dynamic content well. The update frequency is too low, so for instance watching a video over a thin client protocol results in viewing a slideshow of video frames. The big impact factors in this case are the bandwidth and the network delay between client and server.

Other results seen in the lab trials are that when the connection is lost, the image freezes which is harmful for interactive applications. When the connection is re-established, the session continues.

### 4.1.3.2 Comparison between existing thin client protocols

Below, we present a summary of the more detailed experiments included in Appendix 1 of this document.. The protocols that were tested are:

- Citrix ICA
- Sun Ray
- RDP
- VNC
  - Standard
  - Hextile
  - Tight
- FreeNX
  - Over LAN
  - Over ADSL

Measurements have been performed for different use cases:

- Typing scenario: simple office work has been done. Typing a letter in a text editing program (Open Office Writer / Microsoft Word), inserting a figure into the text.
- Browsing scenario: doing an internet search, visiting the website of a local newspaper and reading a couple of articles on it. Reading an email. The sites that were viewed are of a static kind: once loaded the content does not change much of its own (apart from some banners)
- Video scenario (Mpeg4Fullscreen): a video is watched using a media player. In this scenario the content of the screen is highly dynamic: big changes occur very frequently.

#### 4.1.3.2.1 Comparison summary

In this section we present a number of graphs to position the different existing protocols with respect to each other.

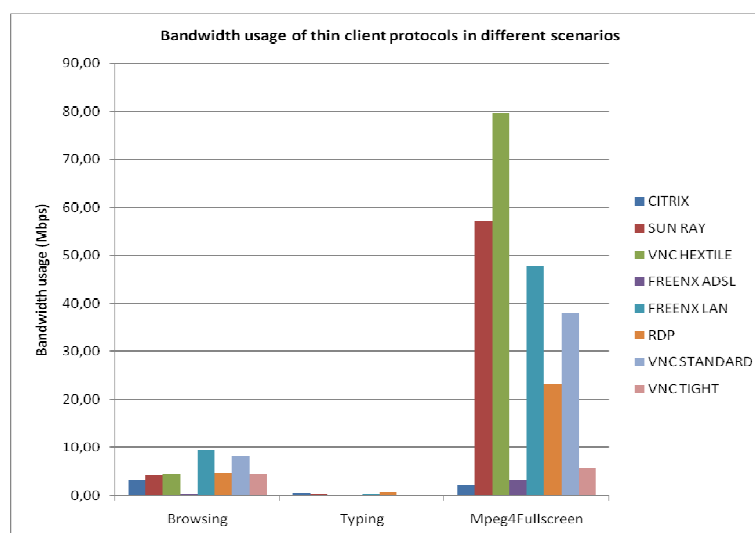


Figure 3 – comparison of bandwidth usage of existing thin client protocols

Figure 3 shows the bandwidth that each tested thin client protocol consumes. In the typing scenario, all protocols perform about equally well, because of the limited amount of updates that are to be sent over the network. Overall, we see that FreeNX over ADSL is the most bandwidth friendly. An important comment on this, however, is that the visual quality was the poorest of all, due to the limited bandwidth that was available. The updates were transported that slow that performing the scenarios was quite hard. In the video scenario we can see that Citrix greatly outperforms the other protocols. In this case another explanation is found: Citrix is able to forward the video directly to the client, which is far more bandwidth efficient (the SpeedScreen Multimedia Extensions were enabled).

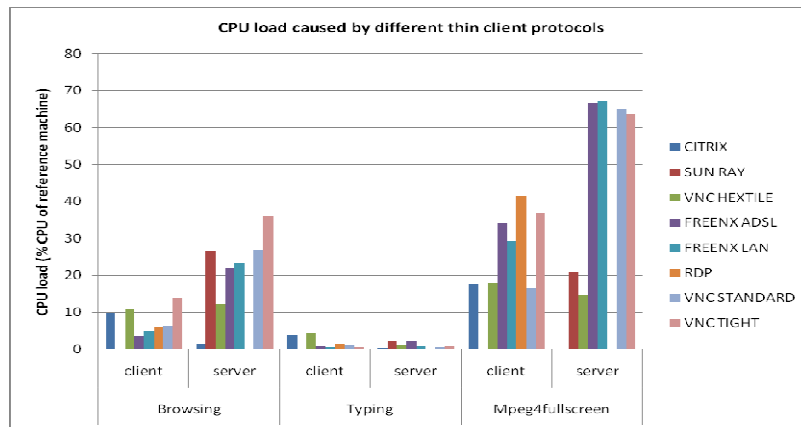


Figure 4 – comparison of CPU load incurred from existing thin client protocols

Figure 4 shows the CPU load incurred by the tested thin client protocols. For Sun Ray, the client CPU load measurements are missing because this protocol operates with a dedicated thin client device that is closed, so no measurements could be performed. For RDP a similar comment applies, at the time of these measurements we were not able to measure the CPU load on the server. For Citrix, the video scenario incurs no CPU load on the server because, as said before, the video was just forwarded to the client for decoding. When looking at the client CPU load, it appears that the optimal solution depends on the scenario. For instance, for browsing, FreeNX over ADSL is the most client-CPU-load-friendly protocol, for more dynamic use cases like the video scenario standard VNC is better. For scalability’s sake, server CPU load is not completely neglectable, and in this perspective Citrix is the better option in all scenarios.

The remark holds again here: these graphs do not have any relation to visual quality.

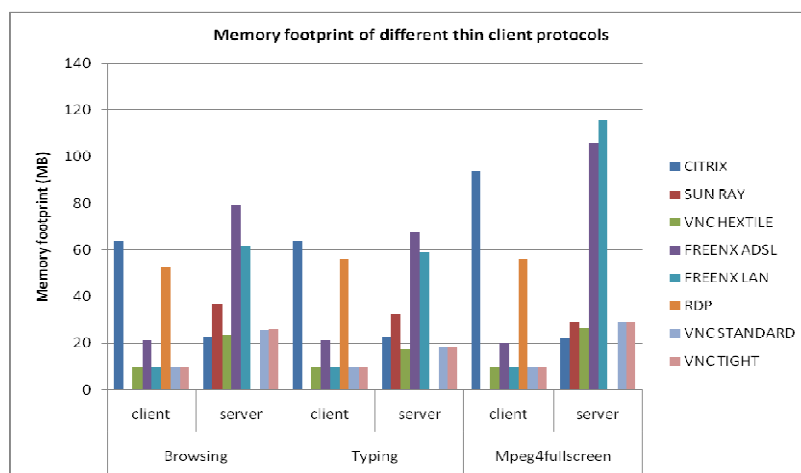


Figure 5 – comparison of memory footprint inherent to existing thin client protocols

Figure 5 compares the memory footprint of the protocols under test. For the Sun Ray client side, no information was available because of the closed nature of the dedicated device. For RDP the same comment as above holds: we were at the time not able to obtain server specific metrics. Citrix appears to require a lot of memory from the client device, possibly because of the video codec that is running used. For the client side, Citrix, RDP and

FreeNX ADSL seem to be memory-unfriendly, the others have about equal characteristics. If memory constraints would appear to be the biggest impact factor in a (probably rare) practical scenario, FreeNX would be the first solution to remove from the list of valid candidates.

#### 4.1.4 Conclusions – requirements met

Important shortcomings of existing thin client protocols (relating to bandwidth consumption and client CPU load) have been found. A demo on these shortcomings (especially in contrast to an optimized system, using a hybrid transmission system, as detailed in section 4.4 of this document), has been given at the 2008 edition of the NEM Summit (as well as during the first review meeting of the project).

For these lab trials, no specific MobiThin implementations have been done to meet requirements.

## 4.2 MPEG BASED IMAGE TRANSMISSION

### 4.2.1 Software/hardware architecture

Under the MobiThin framework, the image transmission is the core issue in designing the appropriate remote display representation able to properly manage the user feedback (user generated events).

The place of the remote display components in the MobiThin architecture is represented in Fig. 6.

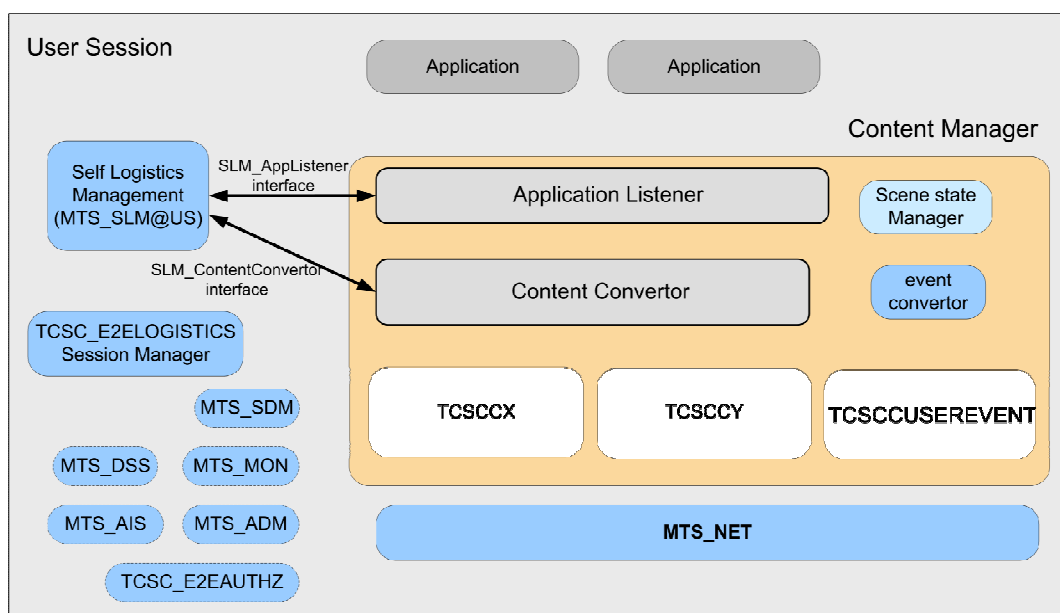


Fig. 6: The remote display as a MobiThin architecture component.

In order to achieve the remote display functionalities for conventional wired thin client environments, several sound technical solutions imposed themselves on the market, like Virtual Network Computing (VNC, [2]), Citrix' Independent Computing Architecture [3] or Microsoft's Remote Desktop (RDP, [4]). However, a lot of improvements are still expected for mobile wireless thin clients. First, the network constraints are far more restrictive, both in terms of bandwidth and reliability. Secondly, the mobile terminal is significantly less complex in terms of both hardware (computing power, graphical facilities, and power autonomy) and software resources. The MobiThin consortium approaches this issue by considering for the first time MPEG-4 based solutions (BiFS [5, 6] and LAsER [7]).

The first *a priori* reasons for this choice are given by their capability to represent in a uniform and heavily compressed way multimedia content, thus alleviating the bandwidth constraints. Moreover, the possibility of installing a low complex player on virtually each mobile thin client terminal strengthens our choice in this respect.

However, several questions are open and cannot be answered without a proper experimental framework:

- *Can the MPEG scene oriented representations offer a good quality for the displayed content?* Note that each and every time a conversion from a content based representation to a light scene based representation is performed, the content loses in quality. Our experiments should bring into evidence whether this quality impairment is acceptable for the applications envisaged by MobiThin (graphics, www browsing, video...).
- *Are the MPEG compression schemes generic enough to work under the frameworks for which they were not initially designed?* Note that although no doubts refer to the power of the MPEG video/audio encoding formats which can be linked to a BiFS/LASer scene, suspicions arise about the effectiveness of the same algorithms applied to content to be displayed on a light terminal in the MobiThin scenarios.
- *Is the interactivity mechanism supported by the MPEG representations generic and quick enough to meet the MobiThin interactivity requirements?*

Note that while the first two questions are in line with the Chapter 7.2.1 in deliverable D.5.1, the last one is new and emerged from the results obtained in the last 12 months. All these questions will be successively answered in the following sub-sections.

#### 4.2.2 Lab trial set-up (hardware description – characteristics)

The logical experimental framework is schematically represented in Fig. 7:

On the server side:

- a Parser (the Application Listener in Fig. 6), developed in MobiThin, will intercept the graphical content generated by the XClient;
- a Converter (developed in MobiThin) will translate it into the MPEG-4 format;
- the GPAC Framework [8] is to encode into an MPEG-4 format;
- the MPEG content will be streamed by a Live 555 streamer as a part of MTS-NET channel;
- an Event handling block (developed in MobiThin) intercepts the user interaction information, captured from the ThinClient.

On the Terminal side, a player will display the content and will intercept the user interaction:

- as a display, Osmo4 player (part of the GPAC framework) is used;
- the user interaction is intercepted at sent through the back channel as AJAX http requests.

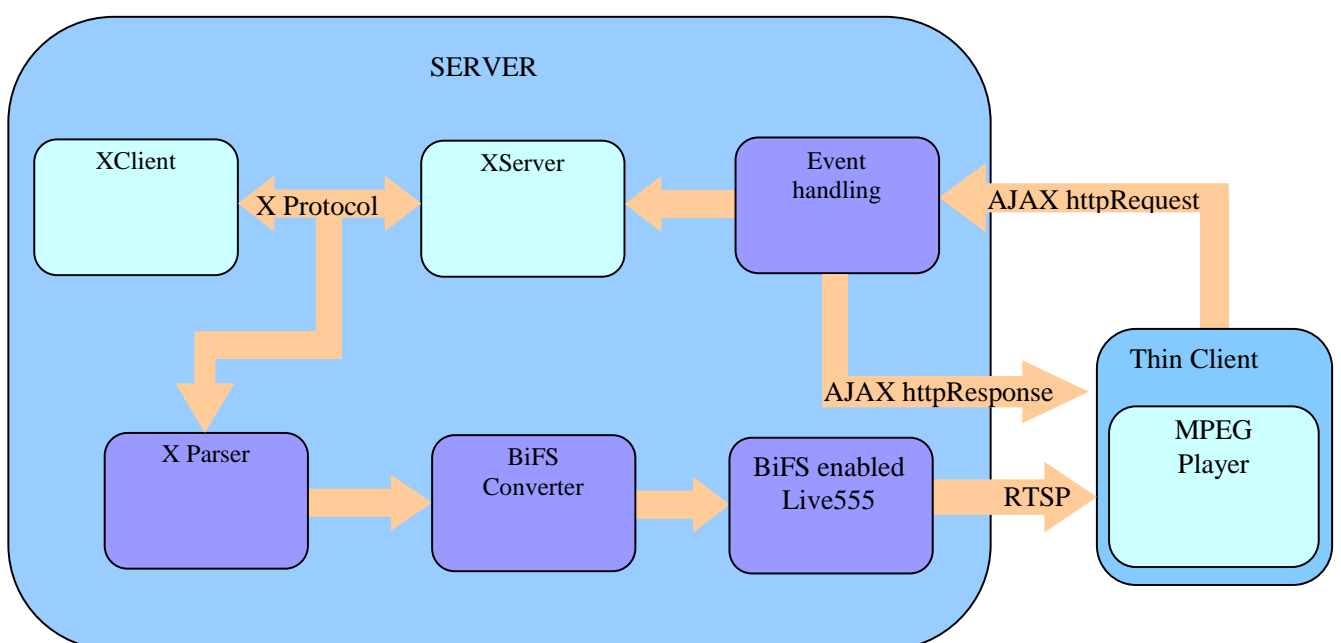


Fig. 7: The logical experimental framework.

From the hardware point of view, a desktop PC (Dual 2 Core 2.8GHz, 2MB of RAM, nVIDIA graphic card) was used. On this system, two virtual machines are running. The first one is Windows XP installed and represents the ThinClient while the second one is a Linux X Server installation (without GNOM - the graphical user interface) and represents the XClient.

The software components presented in Fig. 7 are detailed below.

#### *The X parser*

This application parses the X protocol between the X Client and X Server (both running on the server side) in order to extract the graphical primitives to be rendered. Both X Server and X Client are unaware of this parser. By opening a socket to which the X application is connected, the parser intercepts the X11 traffic, performs the parsing and passes the information to the BiFS converter.

#### *The BiFS converter & encoder*

Every single X11 request intercepted by the parser is mapped to a function which converts it into the BiFS representations. The list of X11 graphical request to be converted in the first Phase was imposed by the applications selected for the demo (i.e. all the graphical requests generated by the XClock, “Hello world” and gedit were considered); some examples:

CreateGC, PolyFillRectangle, PolyRectangle, PolySegment, FillPoly, PolyLine, CopyArea, PolyText8, CreateWindow, ConfigureWindow, PutImage, CreatePixmap, *etc...*

The binary encoding is done by using the BiFS encoder provided by the GPAC.

#### *The MPEG player*

Under the project framework, the GPAC player, as part of GPAC multimedia solution packet, has been considered. It is supported on Windows platforms (Osmo4/Osmophone on PocketPC) and all platforms with GCC, SDL 1.2 (and wxWidgets 2.5.2 for Osmo4).

The GPAC player supports rendering of 2D graphics formats like SVG, LASER or BiFS and 3D graphics formats like VRML or X3D. The latest version of GPAC is version 0.4.5 where a lot of improvements are made, such as integrated 2D/3D renderer with support for mixed 2D/3D drawing (documents mixing BiFS, SVG, VRML/X3D).

#### *The Event handler block*

Most nodes in an MPEG scene generate events upon receiving events emitted by other nodes (Interpolators, Background2D, all nodes with fields of type exposedField). However, some nodes generate events without receiving events from other node: *TouchSensor, DiscSensor, PlaneSensor2D, InputSensor, ProximitySensor2D*. Some Scripts are to be developed by the user in order to ensure event handling (i.e. to associate server-side commands to the user event captured in the BiFS scene). In this respect, ECMAScript (the root standard of JavaScript) should be considered. Traditionally, when using JavaScript coding, for getting any information from the server or for sending any information to the server, an HTML form has to be made to GET or POST data. Consequently, the client is supposed to call a SUBMIT procedure to send/get the information, to wait for the server answer, and then to have the results loaded into a new scene. As the server returns a new scene each time a user input is detected, it can run slowly thus becoming less manageable. AJAX may be considered as a solution to this problem. It allows the JavaScript to communicate directly with the server, through the JavaScript **XMLHttpRequest** object [7, 9]. In such a case, with an HTTP request, a user can make a request to, and get a response from a web server, and all the operation will be executed in the background.

In the current implementation, the user events are captured by the MPEG sensors which call appropriate ECMAScript functions (user defined). In order to send this information to the server through the uplink channel, the AJAX http Requests are made. At the server side, the *Event handling* listens to these requests and interprets them. Then, according to the MobiThin choice, two alternatives are possible. On the one hand, the *Event*

*handling* may convert the AJAX http requests and post them to the XServer for further processing. The XServer would then ensure the XServer event management, *i.e.* would update the scene and would send back, through the down link, the graphical information for the updated scene. On the other hand, the scene updates may be sent back to the player directly as AJAX http Responses. Up to this moment, the Event handler processes only the AJAX http Requests corresponding to the BiFS *TouchSensor*.

#### *The BiFS enabled Live555*

LIVE555 Streaming Media [10] represents source-code libraries for standards-based RTP/RTCP/RTSP/SIP multimedia streaming. These libraries can be compiled for Unix (including Linux and Mac OS X), Windows, and QNX (and other POSIX-compliant systems). The libraries can also be used to stream, receive, and process MPEG, H.263+ or JPEG video, and several audio codecs. They can easily be extended to support additional (audio and/or video) codecs, and can also be used to build basic RTSP or SIP clients and servers. They have already been used to add streaming support to existing media player applications, such as "Osmo4" (GPAC player), "VLC" and "MPlayer".

The experimental BiFS enabled Live 555 streamer block is built as an independent application. It integrates the streaming support from the LIVE555 Streaming Media and the BiFS graphical content manager provided by GPAC. The input of the streamer is BiFS MPEG-4 content while its output is sent to the thin client through an RTSP port. Up to this moment, the streamer streams the BiFS MPEG-4 files.

### 4.2.3 Results and interpretation

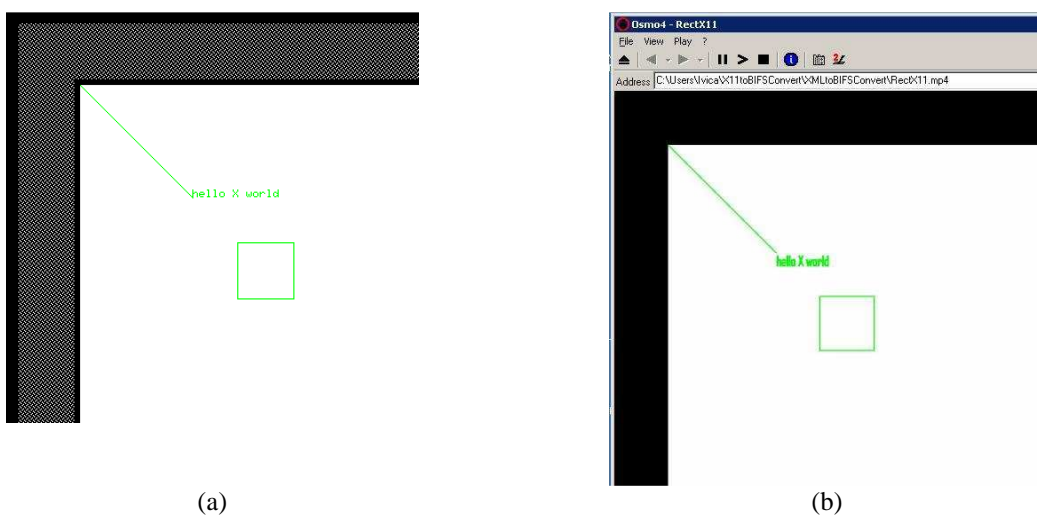
The experimental results are here structured according to the three questions stated in Section 4.2.1:

#### *1) Can the MPEG scene oriented representations afford a good quality for the displayed content?*

In order to check whether the MPEG remote display is able to keep a good level of quality for the render content, three types of applications have been considered up to this moment:

- static scenes composed by basic text and graphics content, Fig. 8.a;
- static scenes combining graphics and images, as generated by the gedit text editor, Fig. 9.a;
- animated graphics scenes as generated by the XClock, Fig. 10.a.

It can be considered that each time the graphical content has been converted without losing in visual quality, Figs. 8.b, 9.b, and 10.b.



**Fig. 8 Conversion of a "Hello world" scene**



Fig. 9 Conversion of the content generated by the Gedit

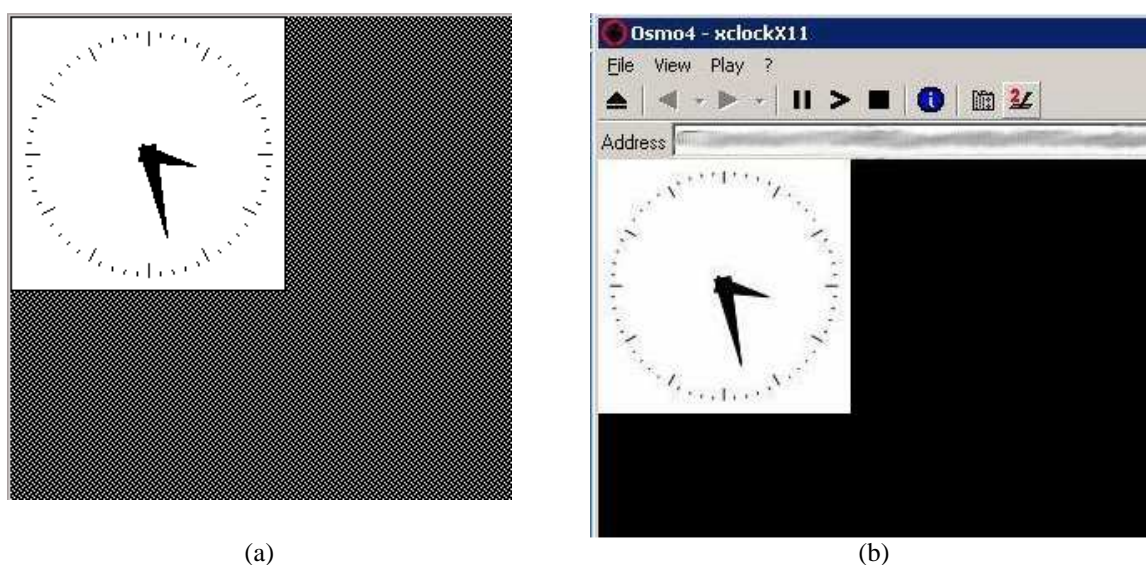


Fig. 10 Conversion of the content generated by the XClock

2) Are the MPEG compression schemes generic enough to work under the frameworks for which they were not initially designed?

In this respect, the experiments focused on the amount of data required in order to represent animated graphics (the XClock), see Tables 2 and 3.

The table below illustrates the amount of bits necessary to represent the XClock in BiFS, as a function of time (the number of minutes for which the clock was running). The data represent accumulated BiFS graphics over certain periods of time, necessary for representing the XClock. It can be noticed that the largest amount of data per time unit corresponds to the initialization (t = 0 min), the rest of the behavior being quite scalable.

Table 2: The data size required for the BiFS graphical content representation.

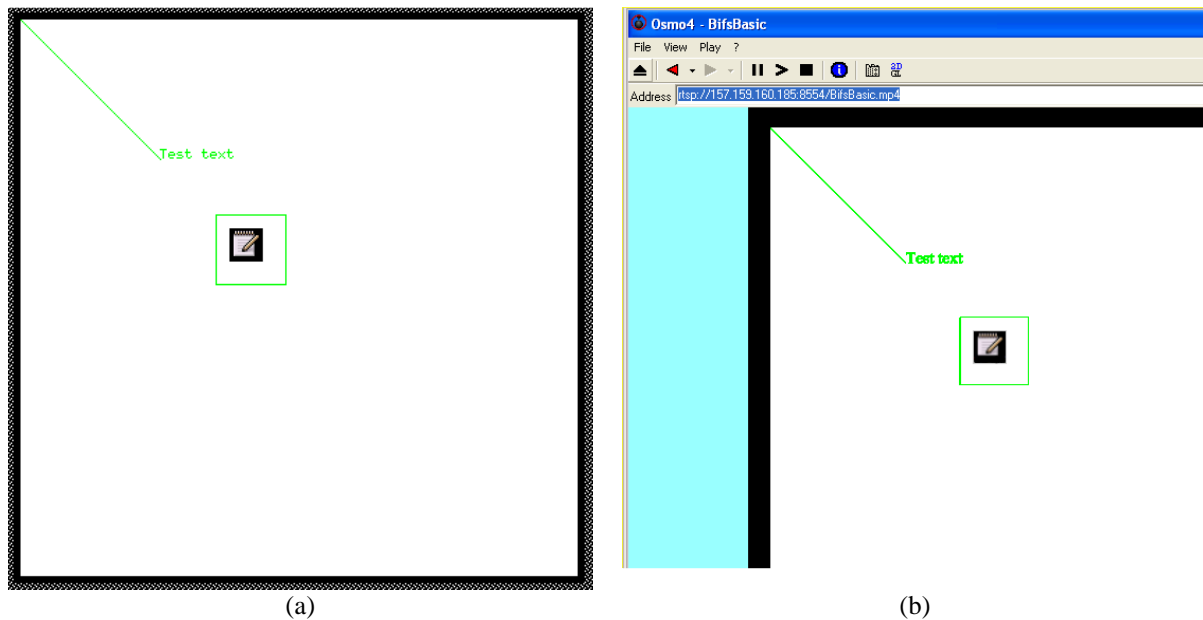
Time [min]	0	1	2	3	4	5	10
Data size [KB]	6.9	26	45	69	90	111	325

We emphasize that the values in Table 1 correspond to “raw data”, i.e. data containing no packetizing information. Table 3 presents a comparison between the traffic generated by the same XClock application in

two cases: (1) when the XLib information is first converted into an MPEG content and then streamed with the BiFS enabled Live555 streamer and (2) when considering the “classical” framework, *i.e.* the Xlib content sent through the VNC protocol. It can be noticed that the MPEG-based representations proved themselves up to 6 times lighter than the corresponding VNC solutions (the value of 6 was obtained by averaging the figures in Table 3). However, the ration between the MPEG and the VNC traffic sizes is time dependent: for short periods of streaming (2-3 minutes), the MPEG is 50 to 10 times lighter than the VNC while for very long periods (e.g. 10 minutes) an average value of about 3.5 was found. Two main reasons may explain this behavior. First, the BiFS compression seems more efficient than the VNC one. Secondly, the BiFS scene initialization is much lighter than the VNC one. This can be also seen when streaming static scenes combining text and graphics, Fig 11: this time, the traffic generated by BiFS (4.5KB) was about 100 times lower than the traffic generated by VNC (492KB).

**Table 3: BiFS vs. VNC traffic.**

Time [min]	0	1	2	3	4	5	10
BiFS [KB]	10	55	98	175	228	245	500
VNC [KB]	594	835	878	1100	1200	1290	1860



**Fig. 11 Text and image content conversion for compression evaluation (original in left and converted in right).**

3) *Is the interactivity mechanism supported by the MPEG representations generic and quick enough to solve all the MobiThin issues?*

The event handling mechanism is illustrated in Fig. 12.a, “open the chocolate” is a text with a *TouchSensor* attached to it. When clicking this text, a specific AJAX http Request is made. This request is interpreted by the *Event handling* and finally the scene is updated (the chocolate is opened) after receiving the corresponding AJAX http Response, Fig. 12.b. The latency of the response depends on the experimental setup (and mainly on the network) but it was always lower than 1ms.

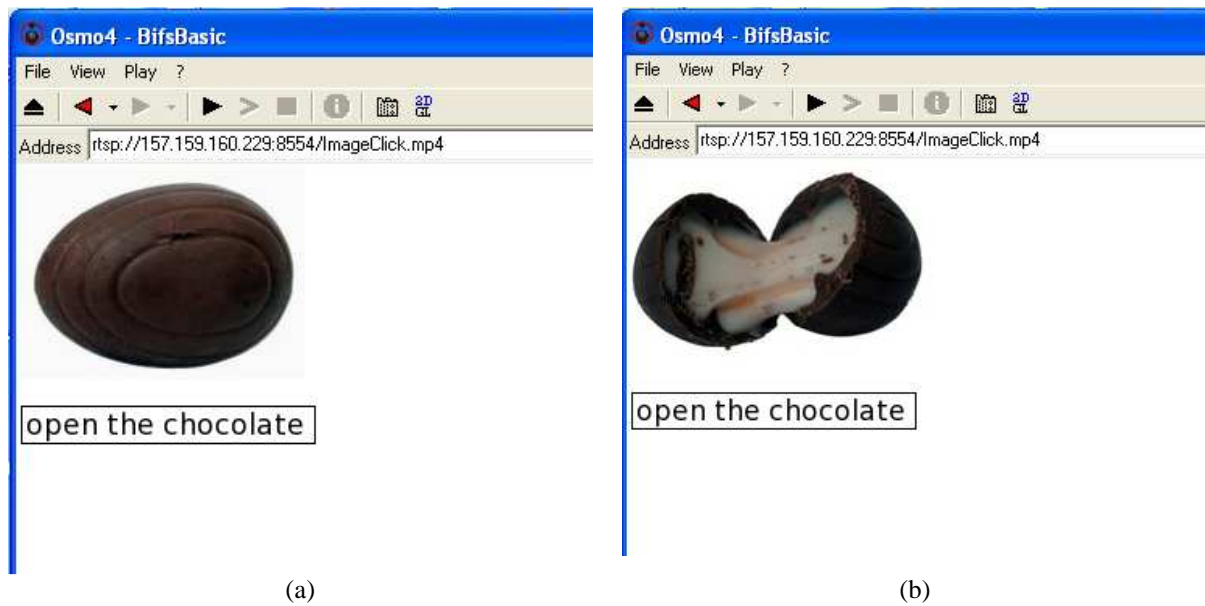


Fig. 12 The “Click” handling

#### 4.2.4 Conclusions – Requirements met

This contribution is meant to be a PoC (Proof of Concepts) for the BiFS appropriateness in remote display graphics solutions. All the experimental results obtained up to this moment were successful: the basic user expectation with respect to content quality and interactivity are met while the traffic is about 10 times lower. Note that all the experiments considered just static and animated graphics and text. Some MobiThin applications (e.g. YouTube-like www browsing) will combine these types of content with some multimedia (video/audio) content. In that case, the video/audio will be directly attached to the BiFS scene and streamed in its native format (without any conversion). In other words, the X11 content should contain a URL to the audio/video which is to be further streamed and played on the player (thin client). Hence, the MPEG advantages will be even more important in such real situations then reported in Tables 1 & 2.

All these preliminary results encourage us to go further in such an approach: the phase 2 of the project will allow the BiFS utility to be objectively evaluated for a larger class of applications, be they B2C (e.g. a www browser) or B2B (e.g. the ABAL application deployed by Prologue). In this respect, reference evaluation of the visual quality and extensive compression evaluation between this solution and classical remote displays are just the first steps.

When looking to these results from the MobiThin requirements point of view, it can be noticed that they address the issues connected to multimedia content representation and compression, as well as those related to the user interactivity; the met requirements are presented in the following table.

No	Requirement	Comment
8	The MobiThin framework SHOULD adapt to the Thin Client device user input interfaces (keyboards, pointing & clicking device, shortcuts...) to increase the range of supported applications.	
13	The mobile device SHOULD support Image, Video, Audio, 2D and/or 3D Graphic codecs.	
14	The Thin Client device SHALL support an embedded display resolution of at least up to 800x600 pixels (at least screen size 3")	
15	The Thin Client device MAY also support an external display resolution of at least up to 1280x1024 pixels (at least screen size 10")	
16	The Thin Client application software SHOULD be efficient in terms of footprint.	

20	The targeted mobile device MAY have a camera	
24	The targeted mobile device MAY provide for a full-fledged keyboard (or pluggable keyboard)	
25	The targeted mobile device MAY provide for a game pad interface	
39	The MobiThin solution SHOULD be in terms of bandwidth equal or better than existing solutions (e.g. VNC, RDP, NX) in similar circumstances while offering at least the same image quality.	

Concerning the MPEG-LASER representations, the work is in progress. The results are not expected to increase the compression performances but rather to trade the complexity to some conversion issues.

Finally, note that the results reported in this section have been disseminated for the MPEG community and are accepted for presentation to an SPIE conference:

- Mihai MITREA, Pieter SIMOENS, Bojan JOVESKI, Bert VANKEIRSBILCK, Abdeslam TAGUENGAYTE, Françoise PRETEUX, *Novel approaches to remote display representations: BiFS-based solution and its deployment within the FP7 MobiThin project*, M-616058, Lausanne - Swiss, February 2009.
- Bojan JOVESKI, Mihai MITREA, Pieter SIMOENS, Iain-James MARSHALL, Françoise PRETEUX, *BiFS-based solution and its deployment within the FP7 MobiThin project: event handling and streaming*, M-16465, Maui – USA, April 2009.
- Mihai MITREA, Pieter SIMOENS, Bojan JOVESKI, Bert VANKEIRSBILCK, Abdeslam TAGUENGAYTE, Françoise PRETEUX, *BiFS based approaches to remote display for mobile thin clients*, paper No. 7444A-14, accepted for SPIE Annual meeting, August 2009.

### 4.3 MOBILE DEVICE INTEGRATION OF A THIN CLIENT

The purpose of this proof-of-concept is to demonstrate the use of a thin client protocol in a real mobile phone.

The objective is to identify the limitations or issues of a basic thin client protocol when it is used over a wireless link, and of the thin client viewer in use in a mobile phone with a screen of a limited size and limited keyboard input device.

#### 4.3.1 Software/hardware architecture

The Figure 13 presents the thin client overall system architecture. For the purpose of this demonstration, a Thin client software is ported on a Mobile phone named Ocean.

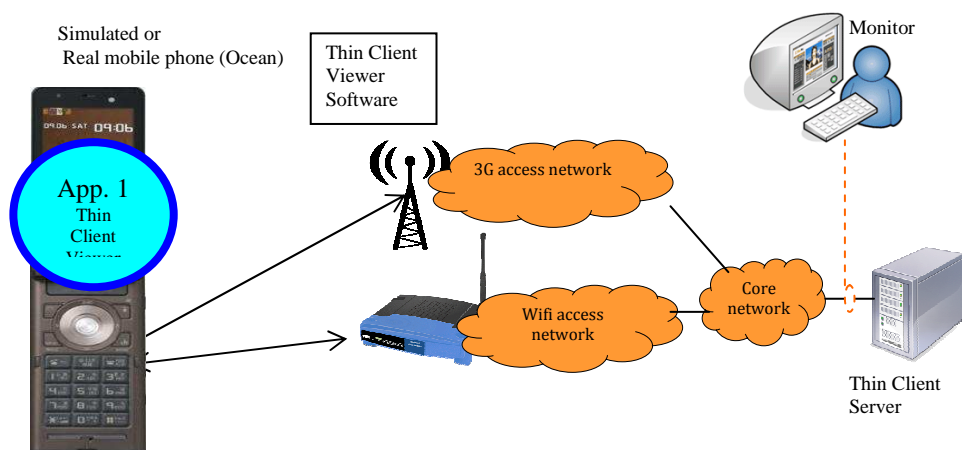


Figure 13 – System architecture

### 4.3.2 Ocean software architecture

The Ocean platform is using a Linux based operating system optimised and adapted to the mobile phone hardware characteristics.

The figure 14 shows the software architecture of the VNC viewer inside the Ocean mobile phone.

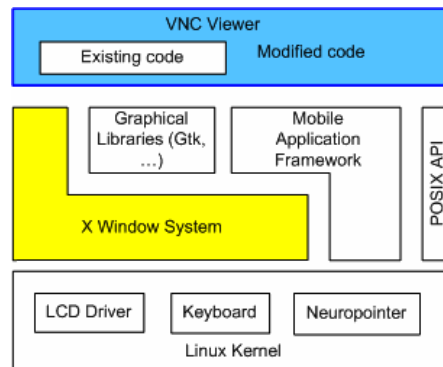


Figure 14 – Software architecture

- The legacy VNC source code is based on X Window System
- The Mobile Application Framework provides specific application programming interfaces for:
  - Registering and controlling the application
  - Co-exist with other mobile applications
  - Receiving keyboard and pointer related events
  - Internal modifications of the application required some architecture re-engineering work

#### 4.3.2.1 Porting of the VNC viewer software on the Ocean Mobile Phone

The Open source code of the Real-VNC viewer was ported on the Ocean platform.

Porting of the thin client viewer involved the following development steps:

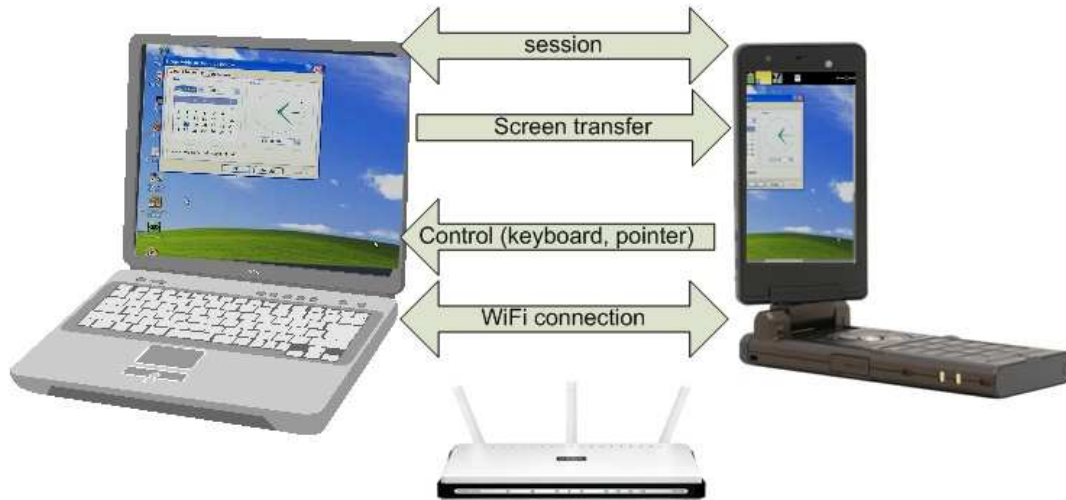
- Setup of the simulator environment (simulator platform build, source code management, ...)
- Port, adapt, integrate, and validate a VNC viewer application on simulation environment.
- Setup of the target environment (real platform build and development environment setup, Wireless network setup, ...)
- Port and validate the modified application on Target

Several adaptations of the viewer software were needed to adapt it to the specific embedded platform:

- The VNC viewer application has to be registered, and to be controlled specifically by the underlying operating system.
- The viewer application has to Co-exist with other mobile applications.
- The viewer application have to Receiving keyboard and pointer related events

### 4.3.3 Lab trial set-up

The Target environment consist of the real Ocean mobile phone, the wireless network, and the VNC Server as depicted in the next figure



### 4.3.4 Description of experiments performed and the results

We validated the application firstly on the Ocean simulator and then started to perform tests on the wireless network environment.

The test performed were focused on the identification of specific issues of the viewer over wireless link, and on the usability of the VNC viewer on a Mobile terminal that have a limited screen and reduce Keyboard functionality.

#### 4.3.4.1 Test of the VNC viewer on Ocean over a wireless network

The following test of VNC viewer over a wireless network were performed :

- UI tests
- Display update tests
- Performance analysis tests

The VNC viewer on the Ocean behaves correctly over WiFi, in a similar way as a standard PC with a WiFi card would have done.

In terms of bandwidth, the WiFi network allowed a maximum bandwidth of 54Mbit/s. The user tests show that the actually available bandwidth was sufficient to bring a good quality of experience to the User (as in this situation, VNC is transported over WiFi, bandwidth usage is in line with the values reported in section 4.3.1).

It should be noted that the CPU of the Mobile phone is sufficient to run the viewer. The display of the remote PC screen was displayed quickly offering a good quality of experience for the remote user.

#### 4.3.4.2 Usability of the VNC viewer on a mobile phone

We firstly performed some basic connectivity test between the VNC viewer running on the client and the remote PC server.

Regarding the establishment of the session between the mobile device and the remote PC server, the following result are reported:

- The current software requires that the user enters the connection parameters such as the IP address of the server. Also this is a quite trivial operation to do with a standard PC that has a wide screen and a full keyboard, this is not as easy to do on a mobile phone due to the limited capability of the device. The current software remained anyway usable, but the user interface should be improved in commercial thin client viewer software running in order to simplify the user interface. This remark should be applied widely for all the UI of the viewer.

Regarding the usability of the VNC viewer application on a mobile device, the following points are reported:

- The Ocean phone has luckily a pointing device (ie: “Neuropointer”) that is located in the middle of the Left/Right/Up/Down key of the keyboard. It allows moving the pointer on the remote PC similarly as a mouse device. It should be noted that without such a pointing device, the usability of most of the application on the remote desktop would be very poor. The use of a convenient pointing device is seen as mandatory to control a remote PC. It is foreseen that in a short future most mobile phone will offer a touch screen that will certainly give benefits in improving the software a lot the software usability.
- The screen size of the Ocean already supports a screen size of 840x480 pixels. This allows displaying a significant part of the remote PC screen. As current desktops tend generally to offer wider resolution, the user of the VNC viewer on a mobile phone will have to scroll the screen to display the interesting part of the screen. We implemented an automatic smooth scrolling of the screen by moving the pointing device on the edge of the screen to improve the software usability. It is foreseen that for commercial version of the viewer software an easy Zoom/UnZoom feature will have to be implemented to significantly improve the usability.
- In order to enter text easily on a keyboard phone, we interfaced the VNC viewer with a local “text editing” application, that allow to enter and edit a text string. This application is well suited for entering/editing text locally on the mobile by using the limited keyboard. It supports the entry of text in ABC mode, and T9 mode. Despite this application, the entry of text remains quite difficult. An improvement of the text entry/editing by voice recognition or by plugging an external keyboard if possible, could be highly desirable to improve the usability.

#### 4.3.5 Conclusions – Requirements met

During the first phase of the project we succeeded to port and test a real Thin Client viewer (VNC) over a commercially up to date mobile phone. The test was done in line with the expectation defined in the Proof of concept demonstrator definition document (D5.1).

The main results and conclusion to be reported are:

- The CPU and memory capability of the state of the art mobile phone are sufficient to run thin client viewer such as VNC
- In order to reach a good usability of the VNC application several requirements have been identified
  - The Mobile terminal device must have a pointing device
  - The Viewer software should offer a Zoom/Unzoom feature in order to easily navigate inside the remote screen
  - Current mobile phone keyboard are usable for entering/editing small amount of text. It is recommended that a full complete keyboard (integrated or external) or a recognition software are supported for entering important text sequences

No	Requirement	Comment
7	The MobiThin project, while considering various form factor, will focus on smart phones or PDAs and will also use Laptops or desktop for the proof-of-concept demonstration	
13	The mobile device SHOULD support Image, Video, Audio, 2D and/or 3D Graphic codecs.	
14	The Thin Client device SHALL support an embedded display resolution of at least up to 800x600 pixels (at least screen size 3")	
16	The Thin Client application software SHOULD be efficient in terms of footprint.	
19	The targeted mobile device SHOULD support a pointing devices feature (touch screen, stylus, track ball)	
24	The targeted mobile device MAY provide for a full-fledged keyboard (or pluggable keyboard)	Ocean has a software text editing application.
28	The thin client device SHALL support at least one wireless communication interface (e.g. WiFi, UMTS, WiMAX, LTE, Bluetooth...)	VNC client tested in Wifi
30	The Thin Client device MUST have sufficient interfaces to: <ul style="list-style-type: none"> <li>- user input interfaces</li> <li>- display(s)</li> <li>- network interfaces / systems</li> </ul>	
31	The MobiThin framework SHALL be compatible with at least one commercially relevant OS (e.g. MS Windows, LINUX and Symbian).	
33	The targeted mobile thin client device SHALL support a Modem stack for wireless communication interfaces in Section 6.3.2.2.5.	
34	The targeted mobile thin client device SHOULD support the MobiThin protocol suite as defined in chapter 6.4 Protocol Suite	
35	The thin client device SHALL support a voice/video call application in parallel to the thin client application (e.g. VoIP soft phone)	
36	The MobiThin framework MAY take advantage of resident application son the thin client device.	Text editing application is used to enter text
49	The MobiThin session communication protocol SHALL support the transport of audiovisual data in both up- and downstream direction,	
50	The MobiThin session communication protocol MAY be backwards compatible with existing thin client remote display architectures such as VNC, X, RDP, ICA...	
51	The MobiThin session communication protocol SHALL support the transport of events in both down- and upstream direction.	

## 4.4 PROTOCOL ADAPTIVITY – HYBRID STREAMING

### 4.4.1 Software architecture

As illustrated in Figure 15, a thin client protocol transfers user data and screen updates between the MobiThin Client and the MobiThin Server.

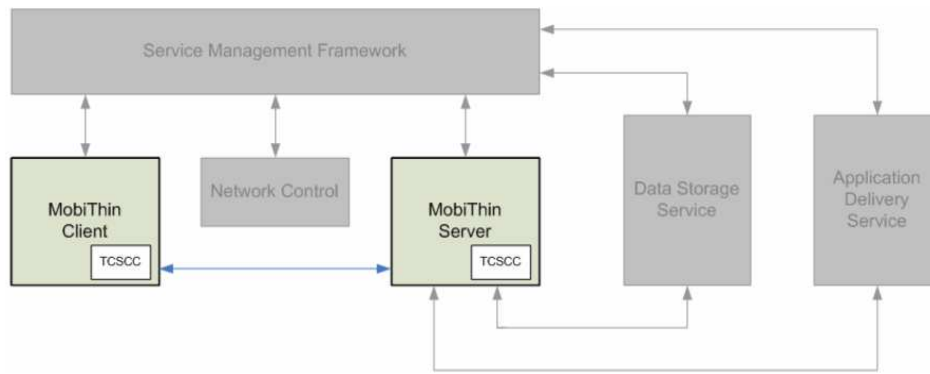


Figure 15 – Protocol adaptivity components involved

In MobiThin, this thin client protocol applies adaptivity as intelligent mechanism to react to environmental changes. This means that management components (such as monitoring), running on the Client and Server, will have a significant role in providing useful parameter data to adapt to.

## 4.4.2 Lab trial set-up

### 4.4.2.1.1 Overall adaptive protocol lab trial set-up

Figure 16 shows the components and devices involved for performing lab trials on protocol adaptivity. As said, the protocol operates between the Thin Client and the Thin Client Server. On the Thin Client machine, the MobiThin Thin Client viewer software must be installed, as well as a monitoring component that senses the adequate environmental triggers that the protocol under test is able to adapt to. For testing purposes, components are to be present that generate events in order to have controlled tests. The same logic is applied to the Thin Client Server, where the MobiThin Thin Client server software is installed, along with a monitoring component, and the necessary software components to perform the tests. Between the Thin Client Server and Thin Client a Network Impairment node is responsible for controlling e.g. the amount of packet loss or delay incurred by the network.

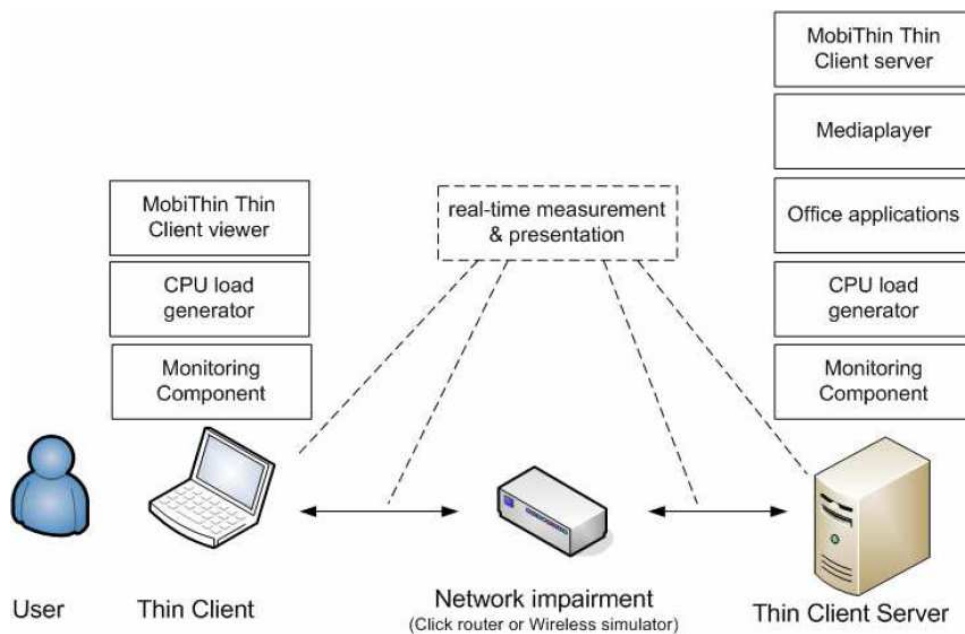


Figure 16 – Detailed view of components involved in protocol adaptivity

The hardware and software details of the servers used for the experiments can be found in the table below.

Table 4 – Hardware and Software details

MobiThin Server	MobiThin Client	Network Impairment
Intel QuadCore 2.73 GHz	Asus eee pc. Intel Celeron M	AMD Athlon 64 2800+ 1800 MHz
2 GB RAM	512 MB RAM	512 MB RAM
1 Gbps NIC	WiFi + 10 Mbps LAN	6 x 1 Gbps NIC
Linux Ubuntu	Linux Debian	Linux Debian
Thin Client Software	Thin Client software	Click router software

#### 4.4.2.2 Energy consumption measurement set-up

In order to measure the client energy consumption the power is measured while executing the different thin client protocols on a real device. Different measurement methods are possible, as shown in Figure 17.

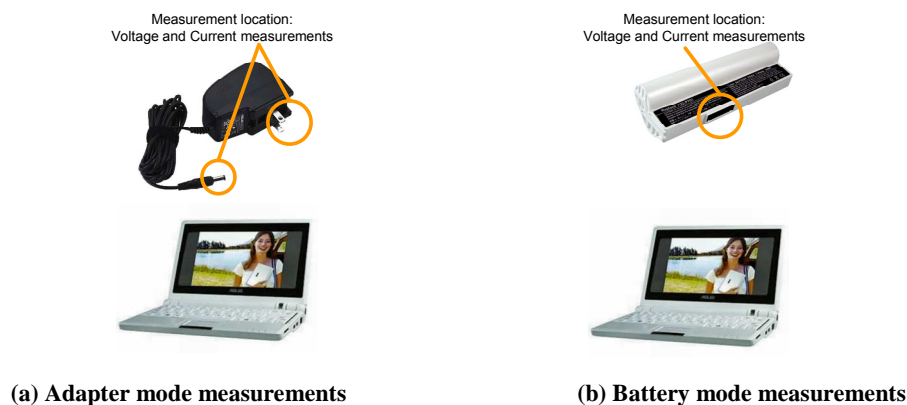


Figure 17 – Energy consumption measurement setups

##### a) Adapter mode measurements

In this set-up, the power adapter is connected to the device and the power that is needed to run the device is measured.

###### 1. AC side

Using a volt and ampere meter, or an electricity (kWh) meter, the power drawn from the adapter is measured at the AC side. These measurements are not that accurate, since the transformation from AC to DC is quite inefficient. The output reading precision of the kWh meter is also not sufficient to measure the power usage of the devices used in this test (1 Wh).

###### 2. DC side

At the DC side of the adapter, more accurate measurements can be performed because the AC/DC conversion does no longer disturb the measurements. The battery is removed from the device, as charging the battery incurs additional power consumption.

##### b) Battery mode measurements

Using the pinouts that were defined empirically by users [11], one could measure battery consumption directly. Practically we were not able to do this because the connectors are not standard and not easily found to buy.

##### c) Software based measurements

Using acpi or other battery performance applications that are available through the operating system installed on the device, the lifetime of the battery could be assessed. However, the problem with this method is that the algorithm behind these applications is not immediately clear, the battery lifetime can differ depending on the initial charge state, and tends to show a (rather non-accurately predictable) non-linear decrease.

Taking measurement precision and practical limitations into account for the device we used in this lab trial, the most appropriate method was the adapter mode measurement at DC side. Both current and voltage are simultaneously measured with a separate Velleman DVM345DI Digital Multimeter.

### 4.4.3 Description of experiments performed

The experiments, or more specifically stated the adaptation mechanisms to be tested for phase I of the MobiThin project, that are identified in deliverable 5.1 are the following:

- Adaptation to graphical content
- Adaptation to network conditions
- Adaptation to client CPU load

In that deliverable, the focus was on demonstrations, so the measured parameters identified there were mainly CPU load and consumed bandwidth. For the lab trials these parameters are important as well, and a measurable important metric that will be added concerns the energy consumption of different adaptive thin client protocols that are developed from within the MobiThin project. For the user, along with visual performance, energy consumption is an important and tangible factor in the overall user experience that is significant as validation of the usability of adaptive thin client protocols.

As already stated before, for the lab trials energy consumption is an important and objective metric related to user experience. Therefore an extra test is included for the lab trials:

- Measurement of the power consumed at client side

### 4.4.4 Results and interpretation

#### 4.4.4.1 Adaptation to graphical content: classic VNC switching to streaming

Results concerning a protocol adapting to the graphical content and the in-depth technicalities of it have been reported in previous deliverables of work package 3. A short summary is given in this paragraph.

The hybrid protocol that switches between classic VNC and streaming mode, bases its switching decision on the analysis of the frame buffer at the server side. This is shown in Figure 18. The application that is executed, calls functions of the local renderer on the thin client server that compute the eventual pixel information that should appear on the screen (through software or using Graphical Processing Unit (GPU) hardware). A component of the thin client protocol analyzes these generated pixels and evaluates how much motion is present in the graphical content (at runtime). If a lot of motion is in the graphical content, streaming is pointed out as the appropriate encoding scheme, if the content is rather static, classic VNC is the protocol to be used. The hybrid encoder then encodes the graphics which are sent over the network to the client.

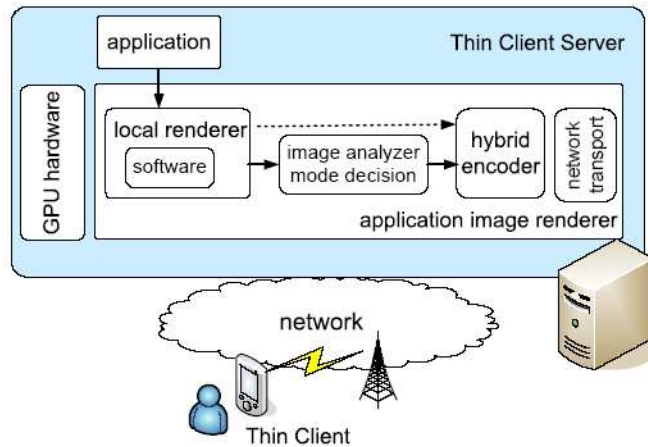


Figure 18 – Hybrid thin client protocol architecture

Figure 19 shows a comparison between encoding the same session using exclusively VNC, exclusively streaming or a mixture between the two by applying the hybrid protocol. In this particular experiment that has been carried out, we started some office work (typing a text and importing a figure in a text editor). Then, at about 20 seconds in the session, a browser was opened and a movie was watched on YouTube: a portion of the screen is occupied with the actual movie playing, and the rest of the screen is static. At 33 seconds, a fullscreen movie was watched. Figure 19 (a) shows the bandwidth consumed by the three protocols under comparison. The VNC curve clearly shows the impact of dynamicity in the screen updates. For streaming only minor variations in bandwidth are observed since at all times the full desktop is streamed from the server to the client. Although in the static case (first 20 seconds of the session), the streaming and VNC solution have similar performance with respect to bandwidth consumption, the hybrid protocol is operating in VNC mode which means that the actual bandwidth consumed at that moment is conform the VNC encoding. At 22 seconds there is a peak in the hybrid protocol bandwidth consumption because of a hysteresis built into the switching mechanism, preventing to switch to fast because of a short-temporal change in mode. Once the hybrid protocol has switched to the streaming mode, the bandwidth consumption then conforms that of streaming.

Although the bandwidth figure shows clear advantage of streaming all content regardless of the dynamic or static nature of the screen updates at the server side, Figure 19 (b) proves that streaming lays a burden on the CPU load at the client side. This is an incentive not to apply streaming for static updates, since in that case the bandwidth consumed is about equal, while the CPU load of streaming is more than 50 times as high.

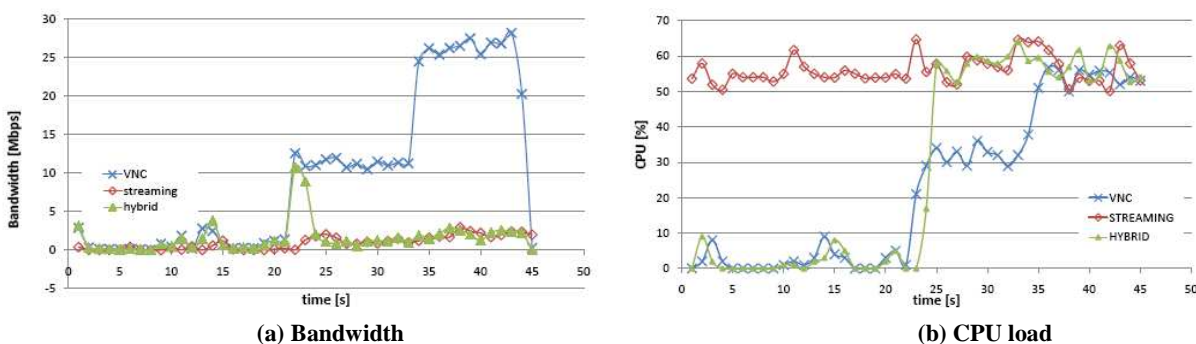


Figure 19 – Comparison between pure VNC, pure streaming and the hybrid approach

Figure 20 shows that the hybrid protocol incurs some server overhead as well. Figure 20 (a) how much CPU load is generated by analyzing the pixels for mode decision only. We have observed an increase of 15% – 25% extra CPU load over encoding the graphic updates, depending on the amount of pixels to be analyzed. The pixel analysis also puts a burden on the end to end delay witnessed by the user. Figure 20(b) shows that the extra delay incurred also depends on the amount of pixels analyzed, but always is of the order of a couple of

milliseconds which is low for a thin client use case, where 80 ms is the maximum end-to-end delay that is acceptable in a gaming environment or more for rather corporate work [12 - 16].

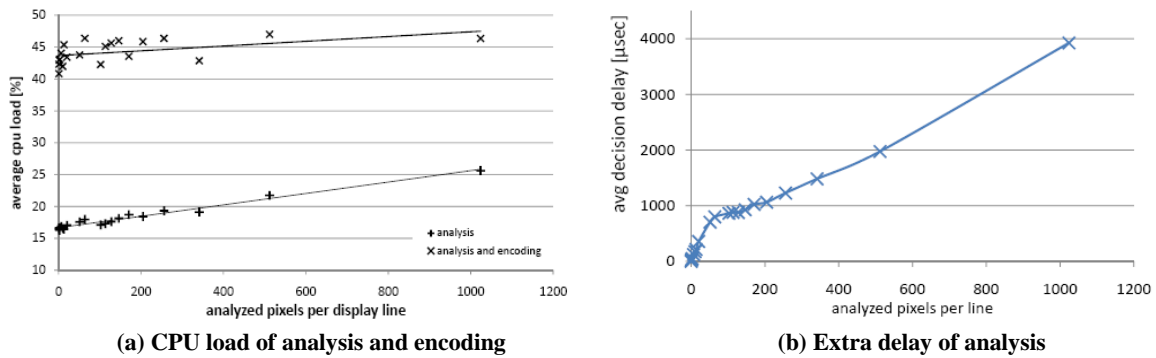


Figure 20 – Hybrid protocol overhead

#### 4.4.4.2 Energy efficiency measurements

The energy measurements performed in the lab trials were done on an Asus EEE PC. Insight into the energy consumption of the device itself was found on the “eee user wiki”, of which the content is replicated in Appendix 2.

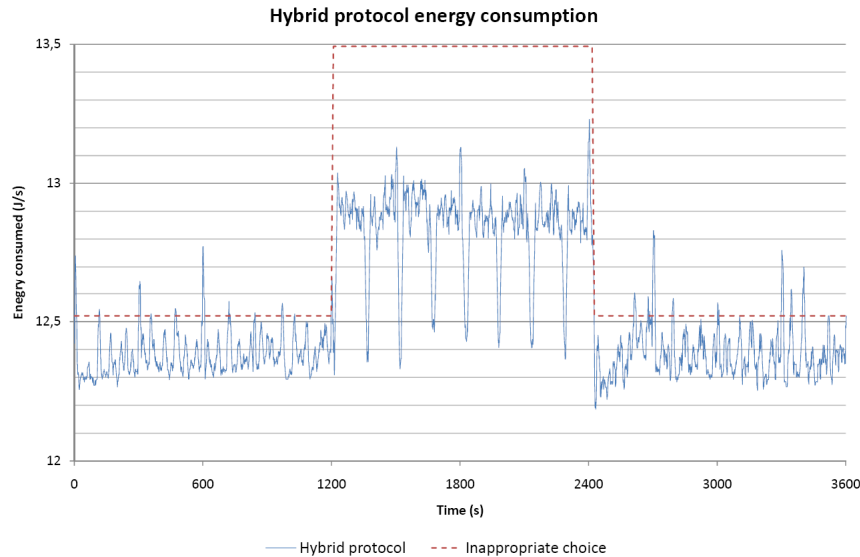
##### 4.4.4.2.1 Energy efficiency of the hybrid (classic VNC – streaming) protocol

Table 5 shows the average energy consumed while executing these scenarios either using VNC [2] as thin client protocol or by streaming the complete desktop from server to client. The table shows that a thin client protocol is efficient for office-type of applications but does not cope with video-type of applications well. Streaming is the better option for dynamic content, but is in comparison with the thin client protocol, not suited for static scenarios. If VNC is seen as the reference encoding scheme, while transferring dynamic content 4% of the consumed energy can be saved by switching to streaming. For static content, VNC is the better option because when streaming 1.6% more energy would be consumed.

Table 5: Average energy consumption of streaming and VNC

	Static	Dynamic
VNC	12.32 J/s	13.49 J/s
Streaming	12.52 J/s	12.95 J/s

Figure 21 depicts how the consumed energy relates to the encoding used for the graphical updates. In this experiment of one hour, the static scenario explained earlier was repeated five times. Every run of this static scenario takes about 4 minutes, resulting in a static period in the trace of 20 minutes. Then, in the same thin client session, the dynamic scenario is executed for 20 minutes, followed by 20 minutes looping the static scenario. The blue, undashed line shows the energy consumed by the hybrid protocol. The red, dashed line shows the average energy that would be consumed if the inappropriate choice is made as explained above, i.e. streaming static content and encoding dynamic content using a VNC as classic thin client protocol.



**Figure 21 – Energy consumption of the hybrid protocol**

This experiment proves that the hybrid protocol chooses the optimal use of two techniques to encode the graphical updates. The switching algorithm is based on the amount of pixels that have changed in a time window of consequent graphical updates. This window has to be kept small enough in order to react to the circumstances in a timely manner: if the user starts video playback the algorithm should switch to streaming quite fast. On the other hand, the time window must be large enough to avoid switching unnecessarily: scrolling a text file while typing ideally should not cause a switch. The settings of the switching algorithm have been defined empirically, but from Figure 21, the big spikes show that for this case the algorithm switched unnecessarily. In the static scenario, a figure is inserted and then the document is scrolled down. This sequence incurs big differences in the screen and took longer than the time window which caused a switch to streaming mode and a switch back to VNC mode shortly after. The spikes in the dynamic period of the test were more appropriate: the video file that was played back contained a closed period of 15 seconds of solid black colour. In this case the algorithm decided correctly that it would be overkill to stream solid colour.

#### 4.4.5 Conclusions – Requirements met

From the demonstrations that were pointed out for use in the lab trials of phase I of the project, some of the protocols that were intended to be implemented were not finished. These were the adaptation to network conditions and adaptation to the client CPU load. On the other hand, an extra lab-trial was identified that takes the energy efficiency into account. Although no large energy savings could be identified by switching between the streaming and the non-streaming mode, important quality improvements were perceived by this switch. The limited gain in energy is due to the fact that the streaming mode reduces bandwidth, but requires more processing power from the terminal device.

No	Requirement	Comment
6	The MobiThin project, while considering various form factors, will focus on smart phones and PDAs and will also use Laptops or desktop for the proof-of-concept demonstration	

10	The Thin client device using MobiThin adaptive protocol SHOULD be energy efficient and be aware of the trade-off to quality and energy efficiency.	
11	The power consumed by a Thin client device using MobiThin adaptive protocol SHALL be lower than when using traditional thin client protocol for the most demanding application *	
12	The power consumed by a Thin client using MobiThin adaptive protocol SHOULD be lower than when using a video-only (such as MPEG-4 V2 or MPEG-4 AVC) based transmission **	
13	The mobile device SHOULD support Image, Video, Audio, 2D and/or 3D Graphic codecs.	
14	The Thin Client device SHALL support an embedded display resolution of at least up to 800x600 pixels (at least screen size 3")	The ASUS EEE PC has resolution of only 800x480 and still shows to be workable
28	The Thin Client device SHALL support at least one wireless communication interface (e.g. WiFi, UMTS, WiMAX, LTE, Bluetooth...)	WiFi is supported in the used device
30	The Thin Client device MUST have sufficient interfaces to: <ul style="list-style-type: none"> <li>➤ User input interfaces</li> <li>➤ Display(s)</li> <li>➤ Network interfaces / systems</li> </ul>	
31	The MobiThin framework SHOULD be compatible with at least one commercially relevant OS (e.g. MS Windows, LINUX and Symbian).	Linux in this case
33	The targeted mobile thin client device SHALL support a Modem stack for wireless communication interfaces in Section 6.3.2.2.5 of Deliverable 5.1.	
34	The targeted mobile thin client device SHOULD support the MobiThin Protocol Suite as defined in chapter 6.4 of Deliverable 5.1 Protocol Suite	
39	The MobiThin solution SHOULD be in terms of bandwidth equal or better than existing solutions (e.g. VNC, RDP, NX) in similar circumstances while offering at least the same image quality.	
47	Protocol parameter SHALL be adapted to the specifics of the traffic generated by the applications	
60	TC server SHALL run Linux applications	
119	TC servers SHOULD have installed the needed libraries and any additional software necessary for MobiThin components.	
120	Ports needed for SMF and TC protocol SHALL be accessible by clients and servers	

\* Similar requirements should appear at the MobiThin system. The power consumption also results from an end-to-end optimization.

\*\* Power consumption is considered globally, including receiver consumption (dependent on the quantity of data received) and decoder consumption (depending on the type of the data, i.e. video or graphics)

\*\*\* Other interfaces can include RJ11 port, PS/2, serial, or parallel interfaces

## 4.5 PROTOCOL ADAPTIVITY - A SCHEDULING APPROACH

### 4.5.1 Software architecture

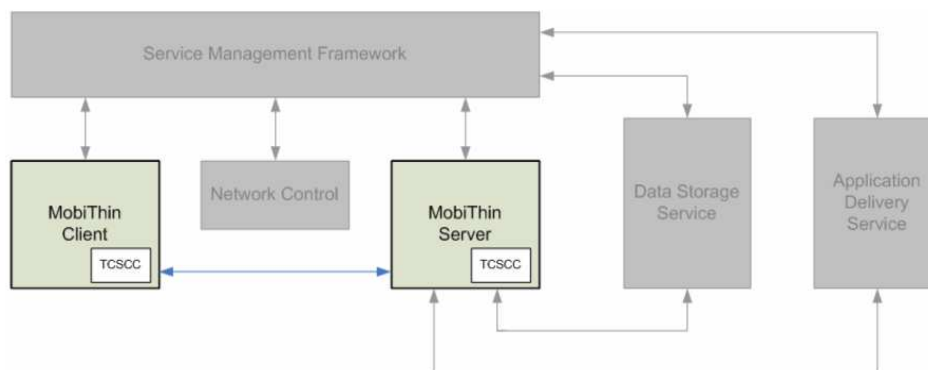


Figure 22– Protocol adaptivity components involved

The thin client protocol transports user data and screen updates between the MobiThin Client and the MobiThin Server, as illustrated in Figure 22.

The thin client protocol can dynamically adapt to the environmental conditions through intelligent mechanisms. In contrast to the previous paragraph (where a switch between streaming and non-streaming was investigated), we here focus on a complementary mechanism, dedicated to the non-streaming situation.

One mechanism is further described in this chapter.

### 4.5.2 A scheduling-based screen update transmission pattern

#### 4.5.2.1 Introduction

The design of adaptive protocol for a thin client system has been identified as a project objective to improve the user Quality of Experience (QoE) and terminal energy consumption over the changing wireless links conditions.

The protocol adaptivity can be done at different layers (application layer, transport layer, link layer) or can adopt cross layer approach to adapt at all layers.

- The protocol adaptivity method that we discussed in this chapter has been done at the thin-client application layer.
- Remote desktop protocols usually follow one of the following display transmission patterns:
  - Polling (e.g. RFB): The client periodically requests an update of the remote application display.
  - Pushing (e.g. RDP): The server autonomously sends display updates to the client.
- We present in the following section a client-driven adaptation mechanism wherein the client requests a specific scheduling for the transmission of application display rendering.
- By using this proposed pattern we expect to minimize the overall bandwidth consumption as well as saving energy.

### 4.5.2.2 Overview

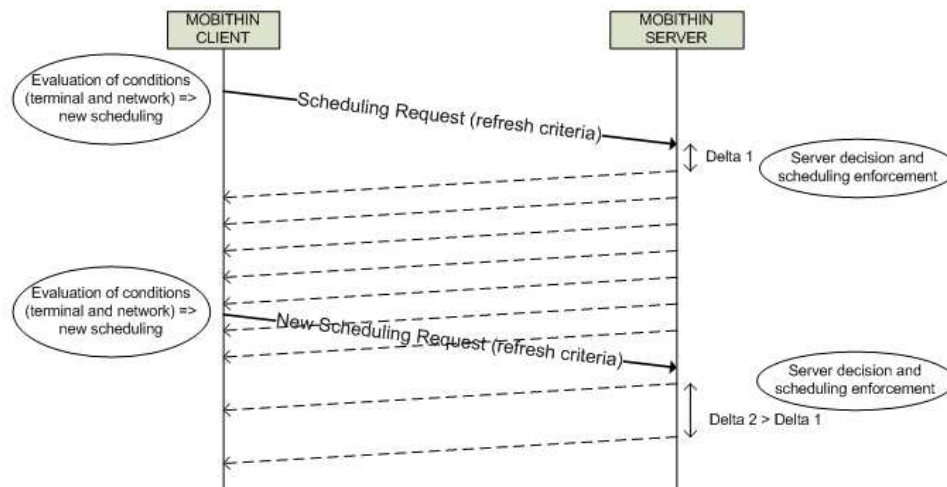


Figure 23 – client-driven scheduled transmission of screen updates

The scheduling-based screen update transmission pattern, as shown in Figure 23, is a client driven method wherein the client evaluates the conditions comprising the terminal capabilities such as the processing speed, the user preferences, or the battery level. Based on the local conditions (network and terminal) the client estimates a scheduling value and sends a “Scheduling Request” message to the server containing specific refresh criteria (such as the area of interest and the desired encoding scheme).

The server receives the Scheduling Request and re-estimate the scheduling value based on local conditions (server capabilities, number of client connections, ...) before enforcing the scheduling of the screen update transmission. Therefore as soon as the server detects some areas having been updated the server transmits the screen data to client based on the the scheduling value.

By using this pattern, the following benefits are expected:

- Adapt the transmission to the client and server capabilities and the varying network conditions,
- Minimize power consumption by adapting the refresh rate to the user real needs
- Optimal use of the available bandwidth:
  - Use the maximum bandwidth if high QoE is expected
  - Use the minimum bandwidth if low QoE is expected.

The following use cases are targeted:

- Office application,
- Internet browsing.
- Video is not a target as other transport protocol and encoding means are more suitable

### 4.5.2.3 VNC-based implementation

To implement the scheduling-based screen update transmission pattern, we modified the VNC protocol. The client can work in either legacy “Client-Pull” mode or new “Scheduled Refresh” mode, as shown in Figure 24 below.

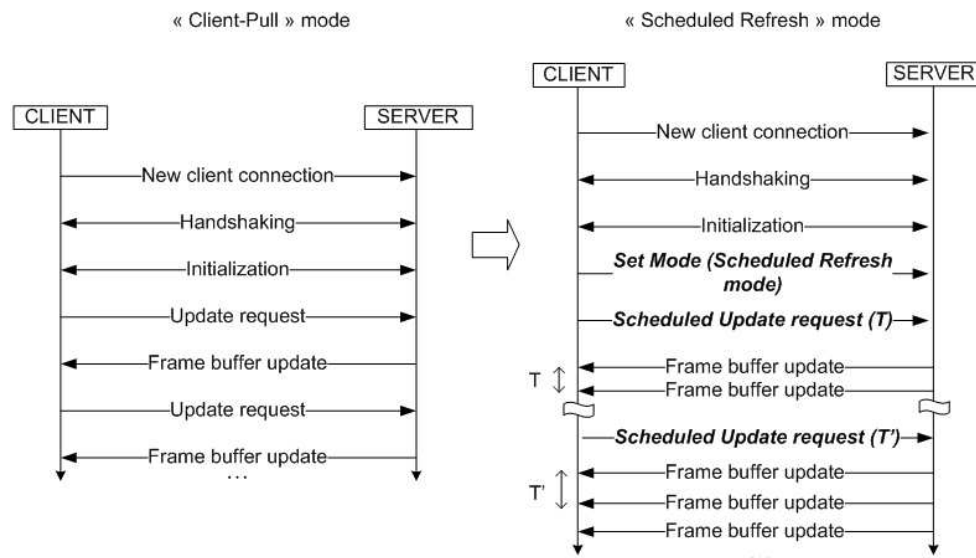


Figure 24 – VNC-based implementation

In Scheduled Refresh mode two new messages were implemented:

- The rfbSetModeRequest message allows the client to dynamically switch between “Client-Pull” mode to “Scheduled Refresh” mode. The server support both legacy and optimized VNC clients. The client can start either in pull or Scheduled Refresh mode depending on whether the server supports the new pattern. To do so the optimized client request an increased number of the protocol version and if the server supports the pattern it sends back the same version number or an older version otherwise.
- The rfbScheduleUpRequest message allows the client to set the period of its screen update. Multiple clients can be configured with different scheduling value depending on their network and terminal conditions.

### 4.5.3 Lab trial set-up

#### 4.5.3.1 Test environment

A test bed was setup in order to compare the optimized VNC protocol with the legacy VNC protocol and discuss the benefits of the scheduling-based screen update transmission pattern. The Figure 25 below shows the network devices and components involved for performing the lab trial on protocol adaptivity.

The Thin Client machines host the Mobithin Client viewer Software which can start either in legacy mode or in optimized mode. The Mobithin Thin Client Server software is installed on the Thin Client Server. Finally a gateway along the path between the Thin Client Server and the Thin Client Device in order to generate some network impairments.

The network impairments for phase 1 mainly comprised the bandwidth shaping. The generation of delay and jitter and potentially data scheduling will be studied in phase 2.

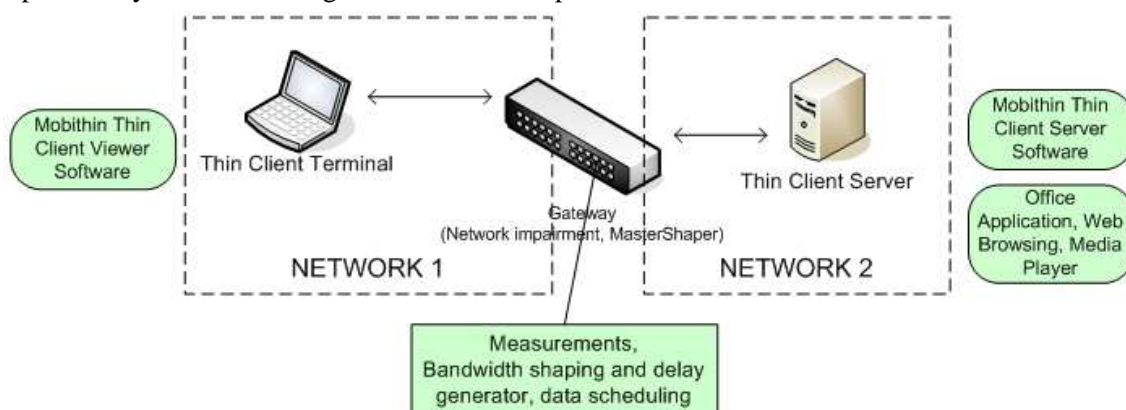


Figure 25 – Test bed environment for testing the scheduling-based screen update transmission pattern

The hardware and software details of the servers used for the experiments can be found in the table below.

**Table 6 – Hardware and Software details**

MobiThin Server	MobiThin Client	Network Impairment
Intel Pentium 4 CPU 2.8GHz	Intel Pentium 4 CPU 2.8GHz	Intel Pentium 4 CPU 2.8GHz
2 GB RAM	2 GB RAM	2 GB RAM
100Base-TX LAN	100Base-TX LAN	100Base-TX LAN
Linux Red Hat Enterprise	Linux Red Hat Enterprise	Linux Red Hat Enterprise
Modified VNC Software	Modified VNC Software	MasterShaper (TC)

### 4.5.3.2 Test methodology

In order to compare the legacy protocol with the new one for different use cases, different scenarios are played for both the legacy client and the optimized client while the measures are collected at the Gateway. The objective of the test was to identify the scheduling value in function of the network conditions and the user scenarios..

The following scenarios were identified:

- office application (text editing, moving windows)

The following parameters were measured:

- The bandwidth consumption

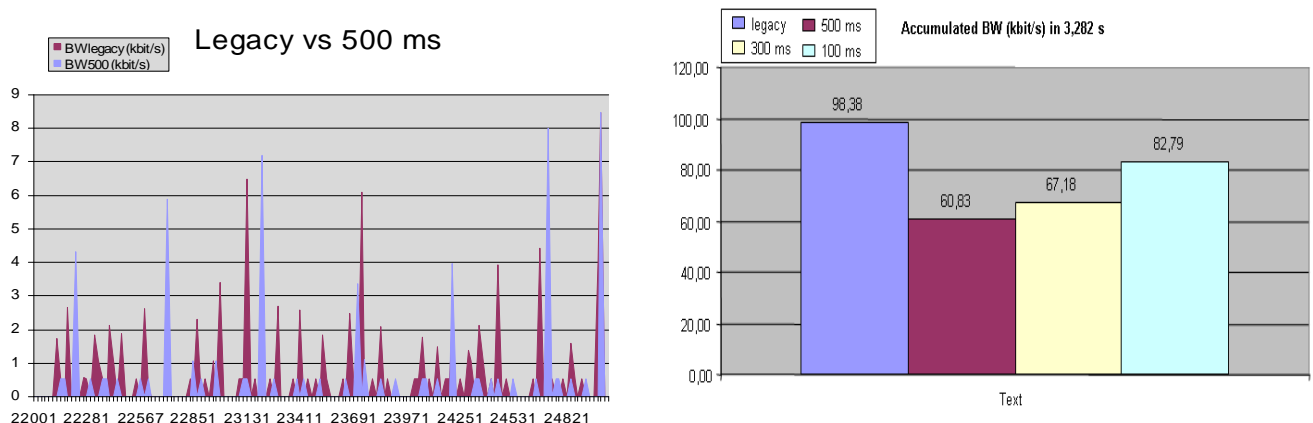
### 4.5.4 Results and interpretation

The following graphics shows the bandwidth consumption for different scenarios:

- Office application: text editing, moving windows,

#### 4.5.4.1 Tests in best network conditions (Local Area Network)

The following figures compare the bandwidth consumed by the legacy protocol with the one consumed by the optimized protocol, during a defined period of time. The use case relates to a user entering some text in an office application and moving some application windows. In this experiment we consider good network conditions as we are on a local area network.



**Figure 26: bandwidth consumption (kbit/s) for text editing use case (left) and its accumulated bandwidth (right). The figures shown relate to different update periods (100 ms, 300 ms, 500 ms), while “legacy” implies the non-optimized situation.**

As we can notice on the Figure 26 above, the overall bandwidth consumption is lower when we use scheduled values (the higher the value the lower the bandwidth consumed). With the scheduled approach the server sends bigger TCP packets and exploits the maximum bandwidth as required instead of sending little packets spread in time. As such the optimized client receives less but bigger updates regrouping all modifications made since the

previous update received by the client. Although we perceived a very good level of QoE with a scheduled value of 100ms, we noticed that 300ms can reduce the bandwidth up to 15% with still an acceptable level of QoE as the latency is barely perceptible.

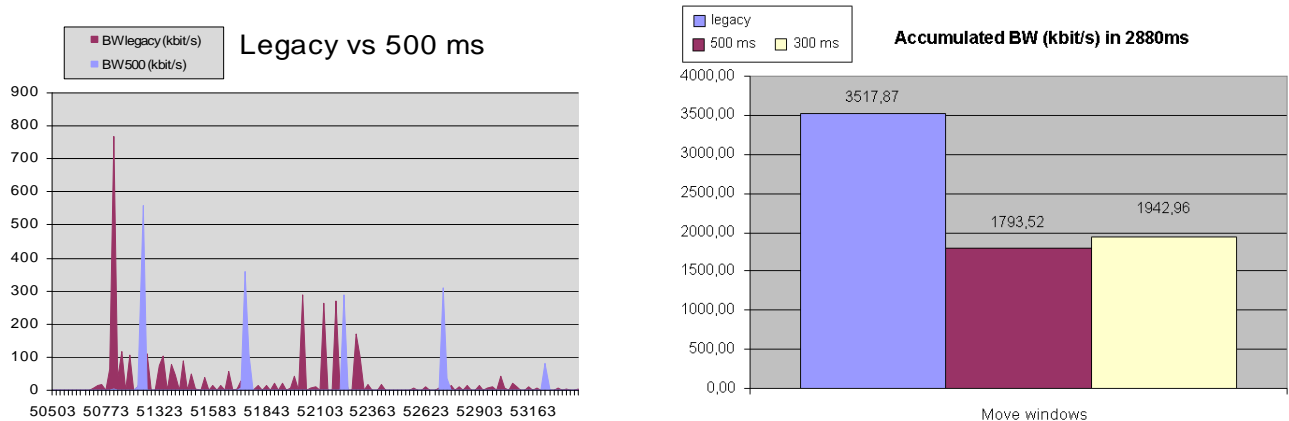


Figure 27: accumulated bandwidth (kbit/s) for the “moving windows” use case (left) and its accumulated bandwidth (right)

The figure 27 comforts our observation made in figure 5 and we notice that the overall bandwidth is also reduced. There is however a slight difference in this use case as the bandwidth can be reduced up to 50%. This is due to the fact that when the user moves a windows, there are much more updates to send to the client, the scheduling approach thereby augmenting the reduction of bandwidth because the higher the screen modification since the last update the lower the bandwidth consumed. Of course this reduction is dependant upon the scheduling value and the expected Quality of Experience. In this use case we found that a scheduled value of 200 or 300 ms was still acceptable in term of QoE.

#### 4.5.4.2 Tests with network impairments (bandwidth reduction)

The following figures compare the bandwidth consumed by the legacy protocol with the one consumed by the optimized protocol, during a defined period of time. The use case relates to a user entering some text in an office application and moving some application windows. In this experiment we consider worse network conditions as the bandwidth is drastically reduced at the gateway.

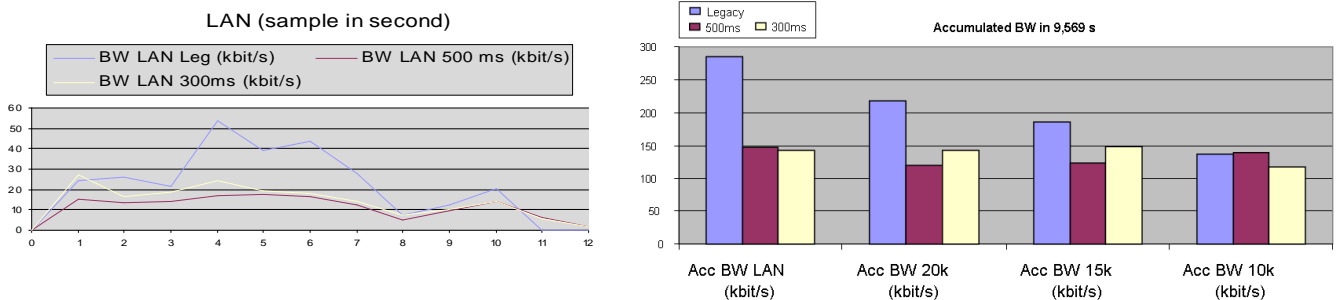


Figure 28: bandwidth consumption (kbit/s) for text editing use case with varying scheduling values (left) and the accumulated bandwidth (right)

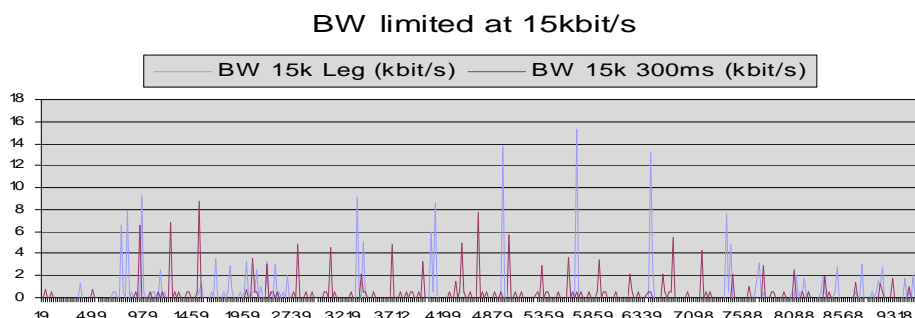


Figure 29: comparison of bandwidth consumption (kbit/s) for text editing use case between 300ms-client and legacy client

Again, as we can notice in Figure 28, the legacy protocol (BW LAN LEG) consumes more bandwidth than with the optimized protocol (BW LAN 500ms or BW LAN 300ms). Moreover by limiting the bandwidth up to 15kbit/s the 300ms-client is not affected by this bandwidth limitation whereas the legacy client starts to lag. Besides In Figure 29 the subsequent peaks for the 300ms-client are well scheduled and there is no lag while we notice serious lags for the legacy-client characterized by jerky peaks of data due to higher TCP retransmissions.

When limiting the available bandwidth to 10kbit/s we noticed that the average bandwidth consumed was slightly the same for the legacy client and the 500ms-client but it is slightly for a 300ms-client. Although there is slight bandwidth gain in this case, the perceived QoE is better for the 500ms and 300ms-client. Therefore the schedule approach allows increasing the QoE in worse network conditions.

Besides there is no big difference between a scheduling value of 500ms and 300ms, expect that the perceived QoE is slightly degraded when using a 500ms scheduling value. So a 300ms scheduling value is good deal in bad network condition where the available bandwidth turns around 15kbit/s.

#### 4.5.5 Conclusions – Requirements met

Based on our preliminary results, we highlighted the following main benefits:

- We can **reduce the overall bandwidth consumption (gain between 15 to 50%)** with acceptable quality of experience. A good deal to still have an acceptable QoE is to set the scheduling value between 200ms and 300ms for office type of application.
- The **scheduled protocol behaviour allows a better quality of experience in worse network conditions when the available bandwidth is reduced.**

In phase 2, we will identify the good scheduling value in order to have an acceptable QoE, reduced bandwidth and power consumption, depending on those varying network conditions. In addition the loop back algorithm will be further elaborated to dynamically adapt to varying network conditions.

No	Requirement	Comment
10	The Thin client device using MobiThin adaptive protocol SHOULD be energy efficient and be aware of the trade-off to quality and energy efficiency.	
11	The power consumed by a Thin client device using MobiThin adaptive protocol SHALL be lower than when using traditional thin client protocol for the most demanding application	
16	The Thin Client application software SHOULD be efficient in terms of footprint.	
31	The MobiThin framework SHALL be compatible with at least one commercially relevant OS (e.g. MS Windows, LINUX and Symbian).	
32	The targeted mobile thin client device SHALL support an adaptive cross layer optimisation feature for bandwidth usage reduction, reduced power consumption, optimal user experience whatever the network conditions.	
34	The targeted mobile thin client device SHOULD support the MobiThin protocol suite as defined in chapter 6.4 Protocol Suite	
39	The MobiThin solution SHOULD be in terms of bandwidth equal or better than existing solutions (e.g. VNC, RDP, NX) in similar circumstances while offering at least the same image quality.	
40	A drop to 20% of the normal bandwidth SHOULD be targeted for the most demanding applications.	For some applications like Office application
46	Protocol parameters SHALL be adapted to device capabilities	
47	Protocol parameter SHALL be adapted to the specifics of the traffic generated by the applications	
50	The MobiThin session communication protocol MAY be backwards compatible with existing thin client remote display architectures such as VNC, X, RDP, ICA...	

## 5. OVERALL CONCLUSIONS

*Text in common for D5.3 and D5.4*

Considerable progress has been reported on the three major parts of the MobiThin system, showing performances in line with expectations. Both in terms of performance and functionality, the approach taken in the project satisfies the requirements put forward in earlier documents. Each of the proof-of-concept reported was designed, observing the architectural requirements (D2.2) and the specified interfaces (D4.1).

Each of the PoC satisfies an important number of requirements, already specified D2.1. Nevertheless, not all requirements are covered (and some are partly covered) by the activities of the first project phase. This information will be used to fine-tune the second phase of the project.

Following reasons why some requirements are not covered by the demo's of Phase I can be identified :

1. The requirement is on the MobiThin environment rather than on the MobiThin system itself (e.g. underlying network infrastructure). As such, these requirements are not to be met by the MobiThin system, but rather by this environment.
2. The work needed to demo that the specific requirement is met, is still in progress. The requirement will be included in Phase II demo's.
3. The component work needed to demonstrate the requirement is Phase II work.

In the table below, a quantitative overview is given to show how many requirements fall within each of the categories. For a limited number of requirements, further discussion will take place in the MobiThin consortium to decide whether to include these in the eventual MobiThin system or not.

Type of requirement	SHALL	SHOULD	MAY	Total
Covered in Phase I demo's	28	21	11	<b>60</b>
Requirement on environment	1	3	1	<b>5</b>
Work in progress, demo in Phase II	29	9	3	<b>41</b>
Component work to start in Phase II	18	9	1	<b>28</b>
Status to be discussed	1	2	3	<b>6</b>
<b>Total</b>	<b>77</b>	<b>44</b>	<b>19</b>	<b>140</b>

**Requirement coverage : quantitative overview.**

More details on these different categories are discussed below.

Regarding the requirements on the MobiThin environment, these mainly address the targeted device (note that many of the requirements on these devices have been included in the demo "Mobile thin client Integration") and the network environment.

21	SHOULD	The targeted mobile device SHOULD have a built-in or external smartcard reader
26	MAY	The mobile device MAY provide for other interfaces such a 1D/2D bar code reader, a RJ45 interface
55	SHOULD	The network layer SHOULD enable terminal mobility

66	SHALL	Data storage servers SHALL allow users to access their data from outside of a thin client session.
67	SHOULD	Data SHOULD be transferred to TC Server at very high speed to avoid any additional latency

**Requirements on MobiThin environment.**

Requirements not demonstrated due to the component work being in progress, mainly relate to:

- The different adaptation mechanisms under study to which the thin client protocol can adapt
- The monitoring/logging subsystem under active development (to be integrated in the SMF shortly)
- Compatibility with MS-Windows (which is likely to be met due to the virtualization approach taken, but which is still to be demonstrated).

43	SHALL	In-order delivery of user events SHALL be ensured
44	SHOULD	Information of link parameters SHOULD be available
45	SHALL	Protocol parameters SHALL be adapted to link parameters
48	SHALL	The MobiThin session communication protocol SHALL support both traffic directly related to the applications and signaling and management traffic
53	SHOULD	The MobiThin framework SHOULD have sufficient signaling capacities in order to manage the two classes of communication (session and server communication).
54	SHALL	The choice between in- or out-of-band signaling SHALL be thoroughly studied.
59	MAY	TC server MAY run Windows applications
81	SHALL	SMF SHALL have different levels of logging - Production, - Tuning, - Debug
82	SHALL	SMF SHALL log all information needed for billing
83	SHALL	SMF SHALL log all information needed for statistics
84	SHALL	SMF SHALL log protocol comments for debugging
85	SHOULD	SMF SHOULD log management activity (adding, modify, delete, ...)
86	SHOULD	SMF SHOULD log user activity
87	SHOULD	SMF SHOULD log supervisor activity (e.g. hot line support activities)
88	SHOULD	SMF SHOULD allow to log selected monitored information
89	SHALL	SMF SHALL know the topology of different servers involved in the MobiThin deployment

91	SHALL	Monitoring services SHALL operate continuously (24h x 7days). Short interruptions for maintenance operations might be tolerated.
92	SHALL	Monitoring services SHALL consume few resources
93	SHALL	Monitoring services SHALL support alert notification (email)
94	SHOULD	Monitoring services SHOULD support flexible alert notification (SMS, Instant messaging, email, ...)
95	SHALL	Monitoring services SHALL support SNMP
96	SHALL	Monitoring services SHALL define Custom MIBS for standard monitoring tools
97	SHALL	Monitoring services SHALL monitor CPU
98	SHALL	Monitoring services SHALL monitor memory
99	SHALL	Monitoring services SHALL monitor the storage of user data
100	SHALL	Number of users per server SHALL be monitored
101	SHALL	Monitoring services SHALL monitor network availability.
102	SHALL	Monitoring services SHALL monitor total bandwidth and per user bandwidth usage.
103	SHALL	Monitoring services SHALL monitor network latency (for each user).
104	SHALL	MobiThin client connection and disconnection SHALL be monitored
105	SHALL	User login and logout SHALL be monitored
106	SHALL	Monitoring services SHALL monitor CPU for each user
107	SHALL	Monitoring services SHALL monitor memory for each user
108	SHALL	Monitoring services SHALL monitor hard drive for each user
109	SHALL	Settings of the MobiThin TC protocol SHALL be monitored
110	SHALL	Exchanged amount of data SHALL be monitored
111	SHOULD	Exchanged amount of data SHOULD be monitored for each protocols
112	SHALL	MobiThin databases SHALL be monitored
113	SHOULD	Operating systems SHOULD be monitored
114	MAY	Energy suppliers MAY be monitored
115	MAY	User applications MAY be monitored

**Requirements addressing work in progress, to be included in Phase II demo's.**

As can be seen from the table below, following component work in Phase II is needed to meet the referred requirements:

- Interface to external data storage and application image service (DSS, AIS)
- Inclusion of remote peripherals

- Interfacing to business support systems
- Support for session mobility (upgrading of the session managing components)
- Support for reservation and profiling

2	SHALL	The MobiThin framework SHALL support any desired application.
3	SHALL	The MobiThin framework SHALL support different business interaction types (also vice versa): - Customer to Customer - Customer to Business - Business to Business
9	MAY	Thin client device MAY switch between an embedded and external display. In consequence the MobiThin framework MAY adapt to the different characteristics of the displays.
37	SHOULD	Interaction delay target for non-demanding applications running inside the MobiThin framework SHOULD be in average 150ms
38	SHOULD	Interaction delay target for demanding applications running inside the MobiThin framework SHOULD be in average 80ms
52	SHALL	The MobiThin session communication protocol SHALL support the redirection of local peripherals to the server.
56	SHALL	TC server SHALL support session mobility
61	SHALL	Application Image Server SHALL run within a server farm
62	SHALL	Application image servers SHALL provide an image of the application including all specific dependencies
63	SHOULD	Application image servers SHOULD support streaming applications to be able to run an application on a thin client server without downloading the entire image
64	SHOULD	Specific user configuration, extensions SHOULD be stored
65	SHOULD	Application image server SHOULD support application update
70	SHALL	SMF SHALL consider the following types of users : - the back-office users - the users of MobiThin services
71	SHALL	The category of users SHALL be defined by back-office user(s) either for back-office users and users
90	SHOULD	SMF SHOULD use the available resilience mechanisms of the infrastructure.
121	SHALL	The MobiThin SMF SHALL know what applications are available at which location
122	SHALL	The MobiThin SMF SHALL support dynamic application delivery from application image servers to thin client servers
126	SHOULD	SMF SHOULD support terminal mobility

131	SHALL	SMF billing services SHALL select monitored information for billing
132	SHALL	SMF billing services SHALL be able to access logging information for billing
133	SHALL	SMF billing services SHALL be able to export billing data in standards billing format
134	SHALL	SMF billing services SHALL be able to transfer billing data to billing system
135	SHOULD	SMF SHOULD Support more than one billing system
136	SHALL	SMF reporting services SHALL select monitored information for Business Intelligence tools
137	SHALL	SMF reporting services SHALL be able to access logging information for BI tools
138	SHALL	SMF reporting services SHALL be able to export data in standards format for BI tools
139	SHALL	SMF reporting services SHALL be able to transfer data to BI system
140	SHOULD	SMF reporting SHOULD Support more than one BI system

**Requirements addressing components, planned for Phase II.**

The requirements shown in the table below are either difficult to demonstrate (requirement), or should be further discussed in the project if it is suitable to include these in the demonstrators of Phase II.

1	SHALL	The MobiThin framework SHALL support residential and business user.
41	SHOULD	The dejitter strategy SHOULD be adapted based on the intrinsic properties of the traffics
76	MAY	SMF MAY take into account anonymous users
79	SHOULD	SMF SHOULD provide unified authentication for the user
80	MAY	SMF MAY additionally support one or more of the following authentication methods (non exhaustive list): - SSO - Access Control Authentication - Network Authentication ...
128	MAY	Session migration service MAY be suspended

**Requirements currently under discussion for inclusion in Phase II demo's.**

## 6. REFERENCES

- [1] <http://xrdp.sourceforge.net/roadmap.html>
- [2] T. Richardson, Q. Stafford-Fraser, K. R. Wood, A. Hopper, “Virtual Network Computing”, IEEE Internet Computing, 1998.
- [3] Citrix Independent Computing Architecture, <http://www.citrix.com>
- [4] Microsoft Remote Desktop Protocol, <http://support.microsoft.com/kb/186607>
- [5] ISO/IEC JTC1/SC29/WG11 14496-11
- [6] [http://www.chiariglione.org/mpeg/tutorials/papers/icj-mpeg4-si/05-BIFS\\_paper/5-BIFS\\_paper.htm](http://www.chiariglione.org/mpeg/tutorials/papers/icj-mpeg4-si/05-BIFS_paper/5-BIFS_paper.htm)
- [7] ISO/IEC JTC1/SC29/WG11 14496-20
- [8] <http://gpac.sourceforge.net/index.php>
- [9] <http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/>
- [10] <http://www.live555.com>
- [11] <http://eeepc.wikispaces.com/7.+Battery> Information about the battery pinout connections of the Asus eee pc used for testing.
- [12] Niraj Tolia, David G. Andersen, M. Satyanarayanan. “Quantifying Interactive User Experience on Thin Clients”, IEEE Computer, Volume 39 – 3, pages 46 – 52, March 2006.
- [13] Niraj Tolia, David G. Andersen, M. Satyanarayanan. “The Seductive Appeal of Thin Clients”. February 2005.
- [14] Pantel, L. On The Impact of Delay on Real-Time Multiplayer Games. International Workshop on Network and Operating System Support for Digital Audio and Video, 2002.
- [15] Dick, M. Analysis of Factors Affecting Players’ Performance and Perception in Multiplayer Games. Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005.
- [16] A.F. Wattimena, et al. Predicting the perceived quality of a First Person Shooter: the Quake IV-model. 5th Workshop on Network & System Support for Games, Netgames, 2006.

## APPENDICES

### APPENDIX 1 DETAILED COMPARISON BETWEEN EXISTING THIN CLIENT PROTOCOLS

Below, the results of testing and analyzing 7 existing thin client protocols are detailed. The thin client protocols that were evaluated are:

- Citrix ICA
- Sun Ray
- RDP
- VNC
  - Standard
  - Hextile
  - Tight
- FreeNX
  - Over LAN
  - Over ADSL

Measurements have been performed for different use cases:

- Typing scenario: simple office work has been done. Typing a letter in a text editing program (Open Office Writer / Microsoft Word), inserting a figure into the text.
- Browsing scenario: doing an internet search, visiting the website of a local newspaper and reading a couple of articles on it. Reading an email. The sites that were viewed are of a static kind: once loaded the content does not change much of its own (apart from some banners)
- Video scenario: a video is watched using a media player. In this scenario the content of the screen is highly dynamic: big changes occur very frequently.

#### Typing scenario

**Table 6: CPU load and memory footprint in typing scenario**

		Client	Server
CPU load (%)	Minimum	0.3 (freenx – lan)	0.195 (citrix)
	<b>Average</b>	<b>1.62</b>	<b>1.07</b>
	Maximum	4.22 (vnc hextile)	2.22 (sun ray)
Memory Footprint (MB)	Minimum	9.45 (vnc hextile)	16.98 (vnc hextile)
	<b>Average</b>	<b>25.68</b>	<b>33.65</b>
	Maximum	64.03 (citrix)	67.44 (freenx – adsl)

**Table 7: Bandwidth used for typing scenario**

Bandwidth (Mbps)	Minimum	0.081 (freenx – adsl)
	<b>Average</b>	<b>0.352</b>

	Maximum	0.8 (rdp)
--	---------	-----------

### Browsing scenario

Table 8: CPU load and memory footprint in browsing scenario

		Client	Server
CPU load (%)	Minimum	3.5 (freenx – adsl)	1.311 (citrix)
	<b>Average</b>	<b>7.76</b>	<b>21.08</b>
	Maximum	13.697 (vnc tight)	35.96 (vnc tight)
Memory Footprint (MB)	Minimum	9.45 (vnc hextile)	22.78 (citrix)
	<b>Average</b>	<b>25.12</b>	<b>39.34</b>
	Maximum	64.03 (citrix)	79.13 (freenx – adsl)

Table 9: Bandwidth used for browsing scenario

Bandwidth (Mbps)	Minimum	0.398 (freenx – adsl)
	<b>Average</b>	<b>4.946</b>
	Maximum	9.413 (freenx – lan)

### Video scenario

Table 10: CPU load and memory footprint in video scenario

		Client	Server
CPU load (%)	Minimum	16.51 (vnc standard)	0.025 (citrix)
	<b>Average</b>	<b>27.61</b>	<b>42.63</b>
	Maximum	41.42 (rdp)	67.42 (freenx – lan)
Memory Footprint (MB)	Minimum	9.45 (vnc hextile)	22.18 (citrix)
	<b>Average</b>	<b>29.77</b>	<b>51.12</b>
	Maximum	93.65 (citrix)	115.9 (freenx – lan)

Table 11: Bandwidth used for video scenario

Bandwidth (Mbps)	Minimum	2.225 <sup>(2)</sup> (citrix)
	<b>Average</b>	<b>32.163</b>
	Maximum	79.68 (vnc hextile)

<sup>2</sup> We have assumed for the Citrix case that the codec of the video is available on both server and client. This is assumed to be also the case in the mobithin environment.

## A1.1 MACHINE SPECIFICATIONS

Following machines were used to do our experiments. They get a name which is referenced to later in this annex.

Computer name	Processor	CPU clock speed (GHz)
Anderson	AMD Athlon 64 Processor 3000+	1,8
Anderson2	Intel Pentium M Processor	1,86
TC96	AMD Athlon 64 Processor 3500+	1
TC97-old	Intel Pentium 4	3
TC97-new	Intel QuadCore 2 Quad CPU Q6600	1,6
TC99	AMD Athlon 64 Processor 3000+	2
Duribreux	AMD Athlon 64 X2 Dual Core Processor BE-235	2,11

## A1.2 REFERENCE MACHINE COMPUTATION TABLE

The reference computation table contains multiplication factors for the CPU speed. This is a rough method to enable comparison of results measured on different computers.

TC96 and TC97-old are in bold and coloured because they were chosen to be reference test machine respectively for server-side and client-side measurements.

	Anderson	Anderson2	TC96	TC97-old	TC97-new	TC99	Duribreux
Anderson	1,00	0,97	<b>1,80</b>	<b>0,60</b>	1,13	0,90	0,85
Anderson2	1,03	1,00	<b>1,86</b>	<b>0,62</b>	1,16	0,93	0,88
TC96	0,56	0,54	<b>1,00</b>	<b>0,33</b>	0,63	0,50	0,47
TC97-old	1,67	1,61	<b>3,00</b>	<b>1,00</b>	1,88	1,50	1,42
TC97-new	0,89	0,86	<b>1,60</b>	<b>0,53</b>	1,00	0,80	0,76
TC99	1,11	1,08	<b>2,00</b>	<b>0,67</b>	1,25	1,00	0,95
Duribreux	1,17	1,13	<b>2,11</b>	<b>0,70</b>	1,32	1,06	1,00

Example usage of this table: a measurement done on Anderson results in 10.05% CPU load. This would have resulted in a CPU load of  $(10.05\% * 0.60) = 6.03\%$  CPU load on TC97-old. Anderson has a slower CPU than TC97-old, so doing the same test on a faster machine would result in less CPU load.

We have to remark that comparing this way is not completely correct. The difference between AMD and Intel processors for instance can't be accurately contained in a division of the clock speeds of both, since they act in different ways. Despite the inaccuracies, this table gives us a better approximation than doing no recomputation.

## A1.3 BANDWIDTH MEASUREMENT DATA

All data in Mbps.

	Browsing	Typing	Mpeg4Fullscreen
CITRIX	3,19	0,53	2,23
SUN RAY	4,50	0,40	57,10
VNC HEXTILE	4,52	0,17	79,68
FREENX ADSL	0,40	0,08	3,39
FREENX LAN	9,41	0,50	47,78
RDP	4,70	0,80	23,18
VNC STANDARD	8,33	0,18	38,20

VNC TIGHT	4,52	0,17	5,75
<i>minimum</i>	0,40	0,08	2,23
<i>average</i>	4,95	0,35	32,16
<i>maximum</i>	9,41	0,80	79,68

## A1.4 CPU LOAD MEASUREMENT DATA

### A1.4.1 Raw Data

With the term raw data we mean the data as is, measured on the machine itself without recomputation to a reference machine.

All data in % CPU of the actual test machine.

	Browsing		Typing		Mpeg4fullscreen	
	client	server	client	server	client	server
CITRIX	9,56	1,311	3,56	0,195	17,5	0,025
SUN RAY	/	26,65	/	2,22	/	20,81
VNC HEX TILE	10,6	11,92	4,22	1	17,94	14,62
FREENX ADSL	3,5	21,68	0,64	2,2	33,96	66,85
FREENX LAN	4,75	23,23	0,3	0,76	29,075	67,42
RDP	10,05	/	1,99	/	69,04	/
VNC STANDARD	6,17	26,8	1	0,48	16,51	64,92
VNC TIGHT	13,697	35,96	0,45	0,64	36,88	63,74
<i>minimum</i>	3,5	1,311	0,3	0,195	16,51	0,025
<i>average</i>	8,33	21,08	1,74	1,07	31,56	42,63
<i>maximum</i>	13,697	35,96	4,22	2,22	69,04	67,42

The measurements were actually done in this test setup:

	client machine	server machine
CITRIX	TC97-old	TC96
SUN RAY	sun ray	TC96
VNC HEX TILE	TC97-old	TC96
FREENX ADSL	TC97-old	TC96
FREENX LAN	TC97-old	TC96
RDP	Anderson	Anderson2
VNC STANDARD	TC97-old	TC96
VNC TIGHT	TC97-old	TC96

### A1.4.2 Corrected Data

In this subsection we have used the reference machine computation table (A1.2). We chose TC96 as reference server machine and TC97-old as reference client machine.

All data in % CPU load as if measured on reference machine.

	Browsing		Typing		Mpeg4fullscreen	
	client	server	client	server	client	server
CITRIX	9,56	1,311	3,56	0,195	17,5	0,025

SUN RAY	/	26,65	/	2,22	/	20,81
VNC HEXTILE	10,6	11,92	4,22	1	17,94	14,62
FREENX ADSL	3,5	21,68	0,64	2,2	33,96	66,85
FREENX LAN	4,75	23,23	0,3	0,76	29,075	67,42
RDP	6,03	/	1,194	/	41,424	/
VNC STANDARD	6,17	26,8	1	0,48	16,51	64,92
VNC TIGHT	13,697	35,96	0,45	0,64	36,88	63,74
<i>minimum</i>	3,5	1,311	0,3	0,195	16,51	0,025
<i>average</i>	7,76	21,08	1,62	1,07	27,61	42,63
<i>maximum</i>	13,697	35,96	4,22	2,22	41,424	67,42

## A1.5 MEMORY FOOTPRINT MEASUREMENT DATA

All data in MB.

	Browsing		Typing		Mpeg4fullscreen	
	client	server	client	server	client	server
CITRIX	64,03	22,78	64,03	22,9	93,65	22,18
SUN RAY	/	36,84	/	32,67	/	29,05
VNC HEXTILE	9,45	23,46	9,45	16,98	9,45	26,73
FREENX ADSL	21,25	79,13	21,23	67,44	20,08	105,64
FREENX LAN	9,72	61,43	9,699	59,03	9,87	115,9
RDP	52,21	/	56,18	/	56,12	/
VNC STANDARD	9,59	25,81	9,59	18,08	9,6	29,16
VNC TIGHT	9,61	25,9	9,61	18,45	9,61	29,16
<i>minimum</i>	9,45	22,78	9,45	16,98	9,45	22,18
<i>average</i>	25,12	39,34	25,68	33,65	29,77	51,12
<i>maximum</i>	64,03	79,13	64,03	67,44	93,65	115,9

## APPENDIX 2 POWER CONSUMPTION OF THE ASUS EEE PC

Taken from [http://wiki.eeuser.com/hardware\\_power\\_consumption](http://wiki.eeuser.com/hardware_power_consumption) on June 17, 2009.

The power consumption of the Eee PC is strongly dependent on what you do with it. This page shows how much various uses of the 700 series Eee affect the power consumption, as measured through the AC adaptor. The AC adaptor provides 10.2-10.3 V when idle, and 9.9 V at the maximum current of 2.1 A (or 20.8 watts). The battery provides 8.3 V when full, and about 7 V (check this) when almost empty, with a capacity of 5200 mAh or 4400 mAh, depending on the model.

It is straightforward to measure the current draw from the AC adaptor when the battery is removed. The numbers below are measured on a 4G model with a 5200 mAh battery. Windows or Linux does not seem to be a big difference.

State	Current (mA)	Remarks
Off	200	
Standby	250-270	
Battery charging	2090-2110	Off or on does not matter

On, screen off	910	Screen off, wireless off, system fully idle
Screen at lowest brightness	+95	
Screen at medium brightness	+145	
Screen at full brightness	+190	
Wireless	+140 - +275	Depends on amount of data transfer. Broadcasting data takes most.
CPU frequency scaling 1:8	-0	System idle. No effect from changing the clock frequency.
CPU usage	+250	Heavy use of the CPU and RAM memory, but not the SSD.
Playing H264 movie, loud volume	+270	
Speakers on	+0 - +100	Depends on what you hear, not on the volume slider.
SSD usage	+55 - +65	Heavy reading/writing to the flash storage
Kicker	+10	The task bar in the full desktop mode eats power.
CPU and SSD	+250	Overall heavy system usage.
SD card	+5 - +75	Idle or heavy usage (tested on a slow (2 MB/s SD card).
USB-powered hard disk	+160 - +500	Idle - heavy usage.
USB optical mouse (laptop)	+10	Small mouse specifically for laptops.
USB optical mouse (desktop)	+40 - +60	Larger mouse for desktop usage.
USB flash drive	+40	
USB hub	+80	Presumably without attached devices
USB bluetooth	+40	
overclock 75→85Mhz	+30	
overclock 75→90Mhz	+40	
External VGA monitor	+10	

From these numbers, it appears that it is hard to extend the battery life dramatically. If a system is running with 30 percent CPU usage (for example web surfing, but not movie watching), with the screen set at 50 percent brightness, the total power consumption would be around 1130 mA. Reducing the screen brightness as much as possible would only save 50 mA (4.4%). The biggest built-in power consumer is the wireless adapter; switching it on without using it wastes 140 mA (12%). Very heavy wireless usage (2 MB/s upload) requires 275 mA rather than 140 mA.

Interestingly, using Intel's SpeedStep technology to change the CPU frequency has very little effect (you can check the control and status files under `/sys/devices/system/cpu/cpu0/cpufreq/`). When idle, there is no measurable difference in power consumption. At 100% CPU usage, reducing the clock frequency by a factor 8 from 620 to 77 MHz decreases the additional current draw a bit (+250 mA at the normal clock speed, a bit less at lower clock speeds), but also makes the computer proportionally slower. On a system that is mostly idle anyway, it would make very little difference, since the CPU would just spend a bit more time at a lower power, with the same total power consumption as a result.

It is harder to measure the actual current draw during battery usage. The standby mode on a 4G system takes 22 hours to fully drain a new 5200 mAh battery, which suggests 240 mA (1.8 W at about 7.5 V). A powered-on system (idle, screen at full brightness) drains the battery in 3:20 hours (1550 mA, 12 W at 7.5 V). Over the power adapter, the corresponding currents are 260 and 1100 mA (2.6 and 11 W), respectively. When off, there does not seem to be a significant battery drain; the power consumption when on the AC adapter (off or standby) seems therefore to be related to the voltage conversion circuit.

It seems that 2.09 A is the maximum current the 700 series Eee PC will drain. So if you're watching a movie on your Eee from a USB hard disk, with the wifi on, and at maximum brightness, it might take 15-20 hours to charge the battery.